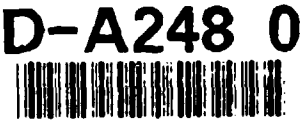


NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

EVALUATION AND IMPROVEMENT OF THE ASW SYSTEM EVALUATION TOOL

by

Peng-tso Chang

March 1992

Thesis Advisor: Yuh-jeng Lee
Co-Advisor James N. Eagle

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/ AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DCLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Computer Science Dept. Naval Postgraduate School 6b. OFFICE SYMBOL (*If Applicable*) CS	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c. ADDRESS (*city, state, and ZIP code*) Monterey, CA 93943-5000	7b. ADDRESS (*city, state, and ZIP code*) Monterey, CA 93943-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 6b. OFFICE SYMBOL (*If Applicable*)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (*city, state, and ZIP code*)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (*Include Security Classification*)
EVALUATION AND IMPROVEMENT OF THE ASW SYSTEM EVALUATION TOOL

12. PERSONAL AUTHOR(S)
Peng-tso Chang

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (*year, month,day*)	15. PAGE COUNT
Master's Thesis	FROM TO	March 1992	

16. SUPPLEMENTARY NOTATION
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17. COSATI CODES			18. SUBJECT TERMS (*continue on reverse if necessary and identify by block number*)
FIELD	GROUP	SUBGROUP	Object-oriented Programming, Object-oirented Simulation, ASW System Simulation, ASW System Evaluation Tool

19. ABSTRACT (*Continue on reverse if necessary and identify by block number*)

The Antisubmarine Warfare System Evaluation Tool (ASSET) is a generic high-level antisubmarine warfare (ASW) modeling tool, designed to aid ASW personnel in the development and refinement of ASW top-level warfare requirements and the ASW Master Plan. The primary objective of this thesis is to analyze and implement the improvements suggested in previous evaluations of various sub-areas of ASSET. The glimpse rate model for submarine detection used in ASSET has been substituted with compound Lambda-Sigma jump model. There is a different target radiated frequency in each environmental region. Each target will have its own detection rate to reflect the differences in its operating characteristics. Multiple engagements between platforms are used to eliminate the limitations of interaction between opponent platforms. The glimpse rate model is used to determine detection opportunities of maritime patrol aircraft (MPA) and to approximate a continuous-looking sensor pattern. A different criterion of selecting search probability area (SPA) and MPA pairs using the ratio of MPA's time on-station over the SPA size was implemented. The feasibility of converting current ASSET code to CLOS was investigated. In addition, part of the code was converted to CLOS.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT [X] UNCLASSIFIED/UNLIMITED [ ] SAME AS RPT. [ ] DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Yuh-Jeng Lee	22b. TELEPHONE (Include Area Code) (408) 646-2361 22c. OFFICE SYMBOL CS/Le

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted SECURITY CLASSIFICATION OF THIS PAGE

All other editions are obsolete UNCLASSIFIED

Approved for public release; distribution is unlimited.

Evaluation and Improvement

of

the ASW System Evaluation Tool

by

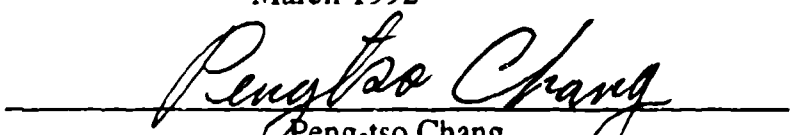
Peng-tso Chang
LT. Commander, Republic of China Navy
B.S., Chinese Naval Academy, 1980

Submitted in partial fulfillment of the requirements
for the degree of

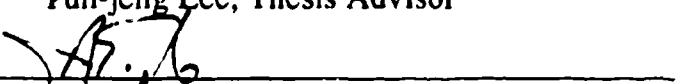
MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
March 1992

Author: 
Peng-tso Chang

Approved by: 
Yuh-jeng Lee, Thesis Advisor


James. N. Eagle, Co-advisor


Robert B. McGhee, Chairman
Department of Computer Science

ABSTRACT

The Antisubmarine Warfare System Evaluation Tool (ASSET) is a generic high-level antisubmarine warfare (ASW) modeling tool, designed to aid ASW personnel in the development and refinement of ASW top-level warfare requirements and the ASW Master Plan. The primary objective of this thesis is to analyze and implement the improvements suggested in previous evaluations of various sub-areas of ASSET. The glimpse rate model for submarine detection used in ASSET has been substituted with compound Lambda-Sigma jump model. There is a different target radiated frequency in each environmental region. Each target will have its own detection rate to reflect the differences in its operating characteristics. Multiple engagements between platforms are used to eliminate the limitations of interaction between opponent platforms. The glimpse rate model is used to determine detection opportunities of maritime patrol aircraft (MPA) and to approximate a continuous-looking sensor pattern. A different criterion of selecting search probability area (SPA) and MPA pairs using the ratio of MPA's time on-station over the SPA size was implemented. The feasibility of converting current ASSET code to CLOS was investigated. In addition, part of the code was converted to CLOS.

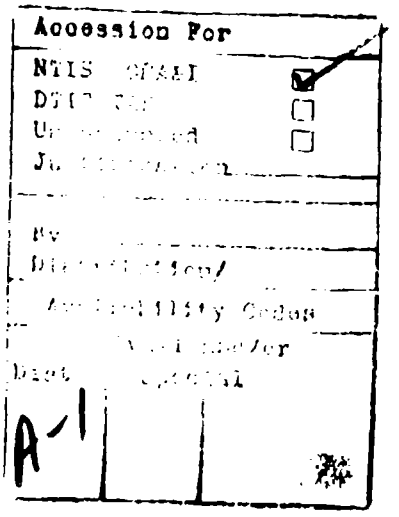


TABLE OF CONTENTS

I. INTRODUCTION

A. OVERVIEW OF ASSET

The Antisubmarine Warfare Systems Evaluation Tool (ASSET) is a generic high-level ASW modeling and analysis tool. It is designed to aid ASW personnel in the development and refinement of ASW top-level warfare requirements and the ASW Master Plan. Current version of ASSET (version 1.0) was implemented in Macintosh Allegro Common Lisp (MACL) version 1.3.2 and designed for use on a Apple Macintosh II computer.

The ASSET is used to model open ocean scenarios involving submarines, maritime patrol aircraft (MPA), shore-based command and data-fusion centers, and a variety of passive acoustic sensors. These tactical objects interact in a particular geographic location against an analogous enemy force structure. The simulation scenario is specified by a user-supplied architecture that determines all aspects of environment, command control, sensor interaction, and platform maneuver. This structure is created through a series of predefined windows, each dealing with a specific topic, such as Geoplot Edit Window, Track Input Window, and Region Build Edit Window.

Once the parameters are determined by the user, the scenario is repeated as a Monte Carlo simulation to produce statistically meaningful measure of effectiveness (MOE). Output data regarding the detection, localization, and prosecution of enemy platforms can then provide a quantitative basis for decisions regarding assessments and system appraisals. The modular nature of

the object-oriented structure of ASSET makes expanding the scope of the simulation possible.

The ASSET is programmed in Macintosh Allegro Common Lisp, an object-oriented programming language, which allows related elements in the simulation to inherit data and functionality via hierarchical class structure. The simulated objects in ASSET can be broken into three groups: (1) command, control, and communication (C3) objects; (2) acoustic radiators; and (3) ASW detectors. The C3 objects consist of level 1 commands, level 2 commands, fusion centers, antisubmarine warfare operational centers (ASWOCs), Submarine Operating Authorities, and communication satellites. A more detailed description of ASSET and its capabilities can be found in the ASSET Technical Documentation and User's Manual [RI 90].

Weaknesses of the ASSET that have been identified include the following: unable to use lower-level models, only part of ASW problem is considered, difficult for the novice to use, excessively simple acoustic and maritime patrol aircraft (MPA) modeling, and slow execution time.

B. PREVIOUS WORK AT NPS

Several evaluations on the various sub-models of ASSET have been conducted at the Naval Postgraduate School. [CA 91] examined and evaluated the acoustic detection modeling in ASSET. An alternate detection model has been suggested that incorporates aspects of the glimpse rate model and the more standard Lambda-Sigma (λ–σ) model.

[SH 91] evaluated the MPA detection and allocation models utilized by ASSET. It also proposed a potential long term course of action for evolving

ASSET into a comprehensive, flexible, and simple-to-use tool for top level ASW appraisals and assessments.

[VE 91] investigated the mathematical development of the tracking algorithm. Some possible modifications to the existing Kalman filter tracking algorithm were suggested to better match the data input and increase computational efficiency.

C. OBJECTIVES

The primary objective of this thesis is to implement and verify the improvements suggested in previous NPS evaluations of various sub-areas of ASSET. We want to test whether the suggestions are feasible. A second objective is to improve the overall efficiency of the ASSET run-time system. The feasibility of converting current Object Lisp code to Common Lisp Object System (CLOS) is also being investigated.

D. ORGANIZATION OF THESIS

Chapter II provides an overview of object-oriented programming language (OOPL) in simulation application. We review the basic characteristics and benefits of OOPL and some applications in the object-oriented simulation field. Chapter III describes the current detection modeling and possible improvements, followed by a description of implementation changes. Chapter IV contains the description of the current MPA modeling, possible improvements, and the details of implementation. Chapter V presents the work of converting current version to CLOS. Finally, Chapter VI is the conclusion and summary along with a brief statement of recommendations for future work. Appendix A contains the source code of revised and new modules of ASSET.

II. OBJECT-ORIENTED SIMULATION

In discrete event simulations, we normally simulate the behavior of various objects that may be mathematical or statistical constructs, or in most cases, physical and observable entities. For example, in a manufacturing factory, the objects are the machines of interest, the types of parts that are to be produced, and the operations needed to complete their manufacture. Such objects are usually the central focus of the simulation studies and are easy to identify. The purposes of simulations are to find a convenient means of modeling the objects and to manipulate and control their behaviors, including the interactions between the objects.

In a simulation, the user creates computer models of the various elements of the simulation, describes how they will interact with one another, and sets them moving. This is similar to typical object-oriented programming, in which the user creates the entities in the universe for the program, describes how they will interact with one another, and finally sets them in motion. Due to the similarities, there are many advantages in using the object-oriented paradigm for discrete event simulations.

A. BASIC CONCEPTS OF OBJECT-ORIENTED COMPUTING

1. Objects, Classes, and Inheritance

Object-oriented languages combine the descriptions of data and procedures within a single entity called an object. An *object* consists of well defined set of variables (called slots within the object) that can be manipulated only by a set of methods defined exclusively for that purpose. A *class* is a

description of one or more similar objects. It is used to specify the implemented data structures and the operations that can be perfonned.

The variables making up an object can be divided into two kinds: class variables and instance variables. A class variable is the one we associate to the class itself and is shared in both name and value by all instances of a class. An instance variable is shared in name only by all instances of a class.

A *method* is a procedure or operation that is defined for an object. A *message* is the instruction sent to an object to perform one of its methods. Ideally, the only way to access any of the variables making up an object is by sending the object a message. In this way, an object is said to be encapsulated in that its internal structure may be modified without affecting user-written code that accesses the object.

Inheritance can be defined simply as code sharing mechanism. It allows a new class to be defined based upon the definition of an existing class without having to copy all the existing code. A child class (or subclass) inherits all the variables and methods defined for its parent class (or superclass). Inheritance is usually transitive, so a class can inherit features from superclasses many levels above. Exceptions can be handled at any level by altering the values of the affected variables.

Inheritance means that the behavior and data associated with child classes are always extensions of the properties associated with parent classes. A subclass must have all the properties of the parent class, and others as well. On the other hand, since a child class is a more specialized (or restricted) form of the parent class, it is also, in a certain sense, a contraction of the parent type.

2. Encapsulation

Data abstraction is a methodological approach to problem solving in programming where information is consciously isolated and hidden in only one part of a program. In particular, the programmer develops a series of abstract data types. Each abstract data type can be viewed as having two faces. From the outside, a client (user) of an abstract data type sees only a collection of operations that together characterize the behavior of the abstraction. On the other side of the interface, the programmer defining the abstraction sees the data variables that are used to maintain the internal state of the object. For example, in a graphics package, a graphical object may be defined to have interval values of position, shape, and extent; on the other hand, operations such as moving and combining objects are all that the user needs to change the internal values of an object. Once declared, the graphical object then becomes a data type within the language. In a data base application, a record becomes a general data type and its operations may include storage and retrieval of the records.

3. Polymorphism

Polymorphism can be defined as allowing a method to operate on objects of different data types. True polymorphism allows the various operations to be created and tested independently of one another. Adding a new version of the operation has no effect upon the existing code at all. Polymorphism allows the definition of flexible software elements amenable to extension and reuse.

Simple polymorphism allows each of different classes to have its own implementation of an operation. Multiple polymorphism allows each class to

have several operations with the same name. The proper operation is chosen based upon the arguments provided.

4. Dynamic Binding

Many benefits of object-oriented programming result from its dynamic binding. When you call a function in a language such as C, the compiler and the linker cooperate to generate a call to a physical address. While this is very efficient, one must take care to associate the function with the appropriate data structures. Strongly typed languages attempt to catch mismatched data types at compile time. Others do a poorer job of catching these errors, and sometimes the problems are not detected until the output starts looking strange.

In object-oriented programming languages, the programmer is relieved of the burden of calling the right method with the specific data structure. Instead, the programmer uses a generic name for the function, and the receiving object looks up for the proper method. This run-time binding is sometimes called "late binding" and has several advantages. In the programming world, late binding means that references are symbolic and methods can be compiled without re-compiling all its callers. The same symbolic names are used, despite the type of object. Finally, a single message can invoke several methods. This is known as polymorphic behavior and it allows code to be written independent of the receiver.

For simulation, late binding means that the specific machine needed for the part can be determined when the part finishes its previous operation. It doesn't have to be established at the start of the simulation. The time to complete the part can also depend on the current set of resources and the state of the system. For example, consider a conveyor handeling packages. How those

packages occupy space on the conveyor is determined at the time they are placed there, not in some predetermined set of fixed-size bins.

B. MODELING PERSPECTIVES

Modeling with objects focuses on what the objects are in the real system, instead of what the objects in a simulation language represent. In a simulated system the general entities can be grouped as classes, and specific entities should be objects. If the objects appear to share some common characteristics, then they can be a subclass of a more general class. Methods should be defined for specific operations if needed. Division of objects into classes, recognition of methods, and the organization of hierarchies form the basic approach to object-oriented modeling.

1. Potential Benefits of Object-oriented Approach

a. Modular Implementation

Object-oriented programming enables the programmer to define an organization of classes that models the relationships among the various kinds of objects. The programer can define classes that serve as building blocks. Each individual aspect of structure and behavior is abstracted and defined separately. The programmer then creates new classes that inherit the desired combination of building blocks. Therefore, object-oriented approach allows for a modular implementation.

b. Conveniently Extensible

The Object-oriented simulation is conveniently extensible. Using a set of classes with documented structure and behavior as building blocks, users can create new classes and add customized behavior. Through function and operator overloading, old symbols take on additional meaning. Inheritance

permits new objects to be defined from existing ones, only the differences need to be noted. Old models are reusable because their methods and objects continue to be useful.

c. Reduced Code Size

In side-by-side comparisons code written in object-oriented programming languages are substantially smaller than that written in procedural languages [CO 87]. The reduced code size means that a single person can manage more tasks. In the simulation of complex systems, building larger and more realistic models are possible without an increase in manpower.

d. Natural Representation of Objects

Objects in most simulations tend to be physical and real, and can easily be represented pictorially. Therefore, object-oriented simulation models often have a natural pictorial (iconic) representation and are easily animated. The user can often translate directly his simulation model into an animated simulation without additional conceptual changes.

Since the objects contain their own functionality, intelligence can be built directly into this functionality using the machinery of artificial intelligence and expert systems. In addition, objects provide a natural basis for concurrency. The idea is that each object could be assigned to its own processor and run independently until some form of coordination is needed. Although it isn't clear exactly what form the coordination should take, there is a natural division among the simulation components when viewed as objects.

e. Reusable Software Systems

By reducing the interdependency among software components, object-oriented programming permits the development of reusable software

systems. Such components can be created and tested as independent units, isolated from other software components.

2. Disadvantages

a. Run-time Cost

Dynamic binding is flexible but with some run-time cost. Dynamic method binding requires the execution of some run-time mechanism to match a method to a message. The cost of dynamic typing involves a run-time search to discover the code to execute every time an operation is used on a data value.

b. Memory

Object-oriented environments (like CLOS) require machines with lots of RAM. In addition, the use of any software library frequently imposes a size penalty over the use of systems especially constructed for a specific project. With the advance in hardware technology, however, this has become less serious.

c. Program Complexity

Although object-oriented programming is often touted as a solution to the problem of software complexity, overuse of inheritance often replaces one form of complexity with another. Understanding the control flow of a program that uses inheritance may require several multiple scans up and down the inheritance graph.

Some object-oriented languages require extensive class libraries be understood before becoming proficient. This increases the learning time and forces users to become more dependent on documentation and high-level debugging tools.

d. No Standard Model

Different object-oriented approaches use different terminology to define similar concepts. There is no single, agreed upon standard for object-oriented model.

C. SIMULATIONS WITH OBJECT-ORIENTED IMPLEMENTATIONS

Many object-oriented systems are suitable for performing simulations. None have clearly been proven to be generally superior as research into understanding object-oriented programming continues. Among the widely used languages, Simula was designed as a simulation language, Smalltalk [GO 83] contains an extensive set of classes and methods to support simulation, and C++ [ST 87] was designed to be applied to simulation. Uses of Smalltalk as a simulation language and environment can be found in [GO 83], and in the tutorial of [KN 86, KN 87].

There are several Lisp-based object-oriented languages with mechanisms for abstraction, polymorphic typing, and dynamic binding. These include Flavors (which supports a more primitive function than an object), CommonLoops, and New Flavors. A Lisp based object-oriented simulation system is described in [ST 88]. DEVS-Scheme is an implementation of DEVS for hierarchical, modular system within an object-oriented framework [KI 87]. DEVS is a simulation formalism developed in [ZE 84].

There are also several hybrid object-oriented systems that combine the object-oriented approach with traditional procedural features. For example, Objective-C [CO 87] adds objects, similar to Smalltalk, to the definition of C. With the similarity between Smalltalk and Objective-C, a translator has been

recently introduced to convert Smalltalk into Objective-C [SC 87]. Thus Smalltalk could be used for design and prototyping, and later converted to C through Object-C for efficient execution. Actor [DU 86] provides a Smalltalk type environment with Pascal-like procedural programming and artificial intelligence features.

Object-oriented systems provide a practical approach for those who design simulation software. Object-oriented languages provide a natural framework for development. The information hiding and abstraction facilities make it easy to develop and maintain complex software components. The extensible platform is an attractive way to add new concepts and features to an existing simulation. It is just a matter of time before existing simulation languages attempt to exploit various aspects of the powerful features of the object-oriented systems.

III. ACOUSTIC DETECTION MODELING

A detection-rate model is used in ASSET version 1.0 to model acoustic systems in which detection opportunities occur over continuous intervals of time. It is used for modeling the detection capabilities of submarines, maritime patrol aircraft (MPA), fixed area sensor (FAS), mine fields, and trip wires.

The first section gives a brief description of the existing ASSET model and its possible problems. The second section reviews the alternate model suggested in [CA 91] and provides some additional ideas on improving the current model. The implementation details for improvement are discussed in the third section.

A. EXISTING ASSET MODEL

1. Glimpse Rate Model

Because of computational limitations, ASSET uses a discrete glimpse rate to model the continuous process of passive acoustic detection. By drawing an exponentially distributed random variable for the time of the next glimpse (i.e., detection opportunity), run time for complex simulations can be held to an acceptable level.

In the glimpse rate model, detection opportunities occur at times determined by a Poisson process with user specified rate g. When an opportunity occurs, the target is detected if and only if signal excess (SE(t)) is greater than zero. The details of glimpse rate model used in ASSET can be found in [RI 90].

The glimpse rate is a user input which greatly affect the results of a run. [CA 91] points out that there is no good guidance for selecting the glimpse rate g. If g is small, the time between glimpses can be quite long. Rapid changes in the

mean signal excess may be missed, causing searchers to have little to no success detecting other platforms. If g is too large, the time between glimpse can become small enough that the model approaches continuous glimpsing, resulting in a unrealistically high cumulative detection probability CDP(t). This happens because ASSET draws an independent fluctuation value for each look, possibly underestimating the expected correlation between signal excess values at closely spaced times.

Another major drawback is that currently there is only one radiated frequency for each acoustic platform in ASSET. In addition, there is a single mean glimpse interval by all searchers regardless of the searcher type or mission.

2. Interaction between Acoustic Platforms

During the period of a hostile encounter, a platform can engage with (or be engaged by) only one opposing platform. If a platform is engaging with an opponent, then it will not be detectable by any other platform.

B. SUGGESTED IMPROVEMENTS TO ASSET

1. Compound Lambda-Sigma Jump Process

The Lambda-Sigma jump process as described in [WA 91] is a model where the times of random fluctuations in signal excess occur according to a Poisson process with rate λ per unit time. The amount of the fluctuation at time t (D(t)) is drawn independently from a normal distribution with mean zero and standard deviation σ. The deviation value remains constant until the next fluctuation occurs in the Poisson process, at which time a new deviation is drawn, independent of previous values. The deviation step function is added to the mean signal excess curve, resulting in a discontinuous signal excess curve. Like the glimpse rate model, a detection occurs if signal excess is nonnegative.

The advantages of using the Lambda-Sigma jump process in ASW analyses are:

(1) It models random fluctuations of signal strength in a handy and adaptable manner.

(2) It captures the idea of correlation in signal excess between closely spaced times.

(3) For simple geometries, close-form expressions for CDP (t) are available.

(4) It is extremely easy to simulate.

a. Global Lambda-Sigma Jump Process

The ocean fluctuation value determined by a Lambda-Sigma stochastic process is used to simulate global area environmental correlation for all targets. This value is used as one of the variables in computing the signal excess.

b. Individual Lambda-Sigma Jump Process

To reflect the acoustic environmental differences among the various searchers' local environment regions, an individual Lambda-Sigma value is also used as one of the variables in computing the signal excess.

c. Total Signal Excess

Total signal excess is the sum of the mean signal excess plus the global and individual Lambda-Sigma fluctuation values.

2. Target's Radiated Frequency

The most detectable radiated frequency of a target will be selected from a list of environment-frequency pairs according to the environmental region it is in. The frequency is then used to determine propagation loss and sweep width in computing the signal excess. This will allow a different target radiated frequency in each environmental region.

3. Target's Detection Rate

Each target will have its own detection rate to reflect the differences in its operating characteristics. Consequently, each searcher-target pair will have its own mean time (between glimpse) for scheduling detection opportunity events.

4. Multiple Engagements between Submarines

When a platform is engaging with an opponent, it is still detectable to other opponent searchers, but it will not detect any more platforms before it finishes the engagement. Thus it is possible for a platform to be detected and attacked while it is engaging a third platform. To simplify the calculation of the probability of a kill in this complex engagement, the events of engagement are sequentially processed. The probability of a kill is treated independently among all searcher-target pairs involved in this multiple engagement. The system will check that either the searcher or target is active before processing engagement events and platform kill events.

5. Responding False Alarm Target

When a false target appears during search, the searcher will send a detection report and will continue the search operations. Also, the platform having a false detection will still be detectable to other unengaged platforms.

C. IMPLEMENTATION OF IMPROVEMENT IDEAS

In this section we describe implementation ideas of the improvements mentioned above. The changes can be classified into three categories: data structures, modified functions, and new functions. They are categorized by their physical locations in the application package distributed by Metron.

1. Structure File

a. PlatformacousticParms

The structure Platformacousticparms specifies acoustic parameters of a platform. The following data fields are modified:

`individualSigma`	Sigma for individual Lambda-Sigma process
`LmabdaSigmaValue`	Individual fluctuation of target's signal excess
`meanInterval`	Mean time between fluctuation
`detectableInterval`	Target's detectable interval (1/Lambda)
`env-freq-list`	List of target's radiated frequencies in each region

2. RB-Installer File

a. New Radio Buttons

Two new radio buttons are added for input of the compound Lambda-sigma jump process:

`lambdaSigmaProcess-rb`	Radio button for input of Lambda-Sigma parameters
`env-freq-rb`	Radio button for input of environment-frequency pairs of a acoustic platform

3. Class Umpire

The class Umpire describes the variables and methods to perform the setup of platform detection and engagement event schedules. It is modified as follows:

a Data Field Deleted

`FOMglimpseInterval`

b. New Data Fields

`globalSigma`	Sigma for global Lambda-Sigma process
`globalLambdaSigmaValue`	Global fluctuation for signal excess
`globalMeanInterval`	Mean time between global fluctuation

c. New Method

`updateGlobalLambdaSigmaValue`: Updates global Lambda-Sigma value and schedules for next update.

d. Modified Functions

`setupForReplication`: Initializes globalLambdaSigmaValue and schedules for next update.

`setNextDetectionOpportunity`: Allows engaged target being detectable to searchers that are not yet engaged. It also checks range when scheduling next detection opportunity for engaged platforms.

`processDetectionOpportunity`: Engaged platforms will not detect any other target.

`setNextDetectionOpportunity`: Allows engaged target being detectable to searchers that are not yet engaged. It also checks range when scheduling next detection opportunity for engaged platforms.

`processDetectionOpportunity`: Engaged platforms will not detect any other target. Engaged platforms is detectable to unengaged searchers.

getDetectionOppResult: Determines target's most detectable radiated frequency. Uses global and individual Lambda-Sigma value to adjust mean signal excess.

processPlatformKill: Checks if the platform had been killed already.

setNextFalseAlarm: Checks if the platform is active before it sets up another false alarm.

processFalseAlarm: Checks if the platform is active before it processes the false alarm.

4. Class EnvironmentManager

The class EnvironmentManager contains the acoustic characteristics information for each environmental region, and defines the methods of getting propagation loss and sweep width. Its modifications are described as follows:

a. Data Fields Deleted

`sigma`

b. Modified Functions

getPL-AN-Sigma: Target's most detectable frequency is used to determine propagation loss and ambient noise.

getSweepWidth: Target's most detectable frequency is used to determine sweep width.

5. Class AcousticPlatforms

The code that describes the variables and operations of an acoustic platform is changed as follows:

a. Data Fields Deleted

`glimpseInterval` Now using target's detectable interval

b. New Data Fields

`individualSigma`	Sigma of individual Lambda-Sigma process
`lambdaSigmaValue`	Individual fluctuation for signal excess
`meanInterval`	Mean time between updating individual Fluctuation
`detectableInterval`	target's detectable interval used to schedule next glimpse

c. Modified Functions

`editEnv-Freq`: Inputs environment-frequency pairs of a platform.

`setForStart`: Initializes env-freq-list, Lambda-Sigma process parameters, target's detectable interval and schedules event for next update of individual Lambda-Sigma value.

`setForStart`: Initializes env-freq-list, Lambda-Sigma process parameters, target's detectable interval and schedules event for next update of individual Lambda-Sigma value.

d. New Functions

`updateLambdaSigmaValue`: Updates Lambda-Sigma value and schedule event for next update of individual Lambda-Sigma value.

6. Changes on User Inputs

This section explains the changes on some necessary inputs that the user needs to specify in setting up the simulation. The details of setting up a whole simulation run can be found in [RI 90]. The user now should enter Global Mean Interval ($1/\lambda$) and Global Sigma during setting up Umpire Parameters. This can be

done by choosing the Edit Umpire Parameters menu item under Simulation menu, which displays the Simulation Umpire Edit window shown in Figure 1.

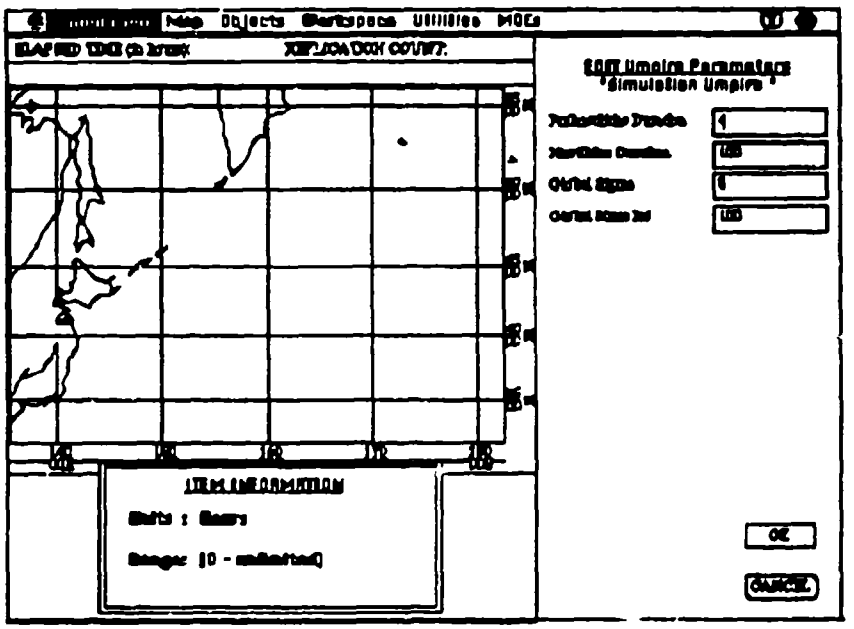


Figure 1. Simulation Umpire Edit Window

The Edit Submarine window in Figure 2 contains added radio buttons: Lambda-Sigma Process, Detectable Interval, and Frequency Environment Pairs. To create or edit a submarine object, the user needs to specify appropriate parameters in Lambda-Sigma Process Edit window shown in Figure 3. For each environmental region the user must enter a most detectable frequency of the platform. The Frequency Environment Edit window is shown in Figure 4.

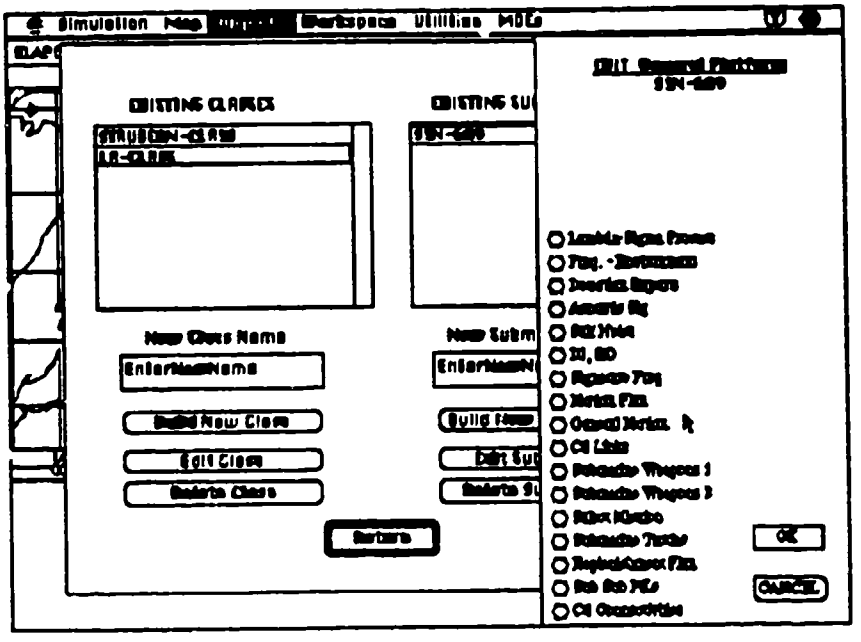


Figure 2. Edit Submarine Window

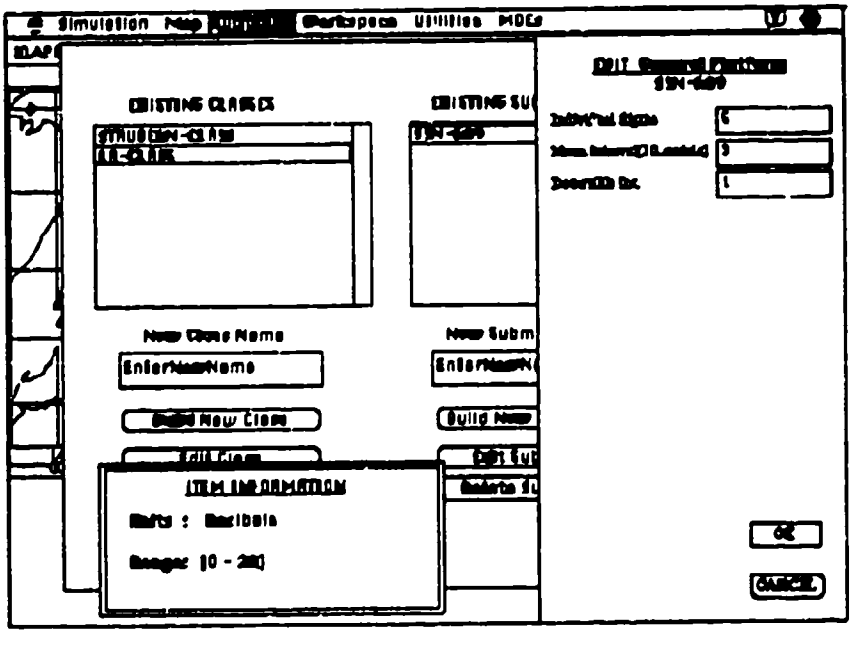


Figure 3. Edit Lambda-Sigma Process Parameters Window

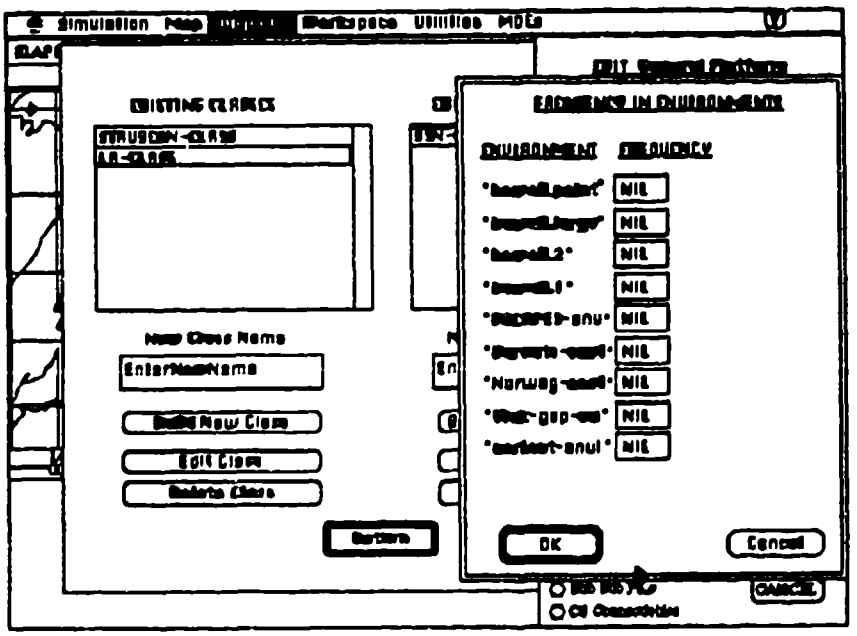


Figure 4. Edit Frequency Environment Pairs Window

IV. MPA MODELING

In this chapter we review ASSET's current MPA detection rate model and allocation model, including discussion of the disadvantages and suggested improvements. This is followed by a description of the implementation details.

A. CURRENT MPA DETECTION RATE MODEL

The MPA acoustic detection model uses a detection rate scheme to decide whether a detection is made. This model is derived from the passive sonar equation and the random search formula.

The MPA is assumed to lay buoys in a pattern to provide uniform coverage of the designated search region. When an MPA is cued to a search region, the search area is the 86% containment region provided by the tracker-correlator. If the MPA is not cued, then the search region has user-input size, and is randomly located in the user-input area search region. If a target is in the designated MPA search region at the beginning of the MPA's time on-station, then it is assumed that the target remains in the search region for the entire MPA's time on-station. And, if the target is outside the search region at the beginning of the MPA's time on-station, then it is assumed that the target remains outside the search region for the entire MPA's time on-station.

1. Detection Rate Calculation

The ASSET calculates a detection rate for each target that lies within the search region when the MPA arrives on-station. The detection rate is the ratio of the area searched per unit time over the total area of the search region. ASSET uses a constant detection rate γ given by:

$$\gamma = NVW / A_S$$

where N is the number of sonobuoy channels processed, V is the average target speed,W is the acoustic sweep width of a single sonobuoy, and A_S is the search area.

The number of sonobuoy channels processed, N, is the smaller of following two values: user-selected number of buoys per search, or the user-selected number of buoys that can be processed.

The single buoy acoustic sweep width, W, is calculated as twice the maximum detection range, R_{max}, the maximum range at which the adjusted or actual figure of merit (FOM) is equal to the propagation loss from the user-entered Proploss Table. Mean FOM is given by:

$$FOM = SL - NL + DI - DT.$$

where SL is the target radiated source level, NL is the total noise level (self-noise + ambient noise), DI is directivity index of the receiver, DT is the detection threshold or recognition differential.

Actual FOM is obtained by adding a value representing the environmental uncertainty correction to the mean FOM. The uncertainty correction is a normally distributed random variable with a mean of zero and a standard deviation of 9 dB. ASSET generates a single uncertainty correction, and thus a single R_{max}, that is used for the entire MPA search period. SL, NL, DI, and DT are user-entered parameters for the particular MPA-submarine pair.

2. Determination of Detection

After the detection rates are calculated for each submarine that is within the search region when the MPA arrives, ASSET sums each of these rates with the user-entered false alarm rate to obtain a collective contact rate (Γ). The

probability distribution for the time to initial contact is an exponential distribution with a rate equal to the collective contact rate. ASSET generates an exponential random number t that represents the time when the MPA detects a real or false target. The exponential random number is obtained from

$$t = - \ln(U [0, 1]) / \Gamma$$

where U[0,1] is a uniform random number between zero and one. A detection is reported at time t if t is less than or equal to the total time that the MPA is on station.

To determine if a target in the search region is detected, ASSET stacks the detection and false alarm rates end to end. ASSET then draws another uniform random number, on the interval [0, Γ], to determine whether a submarine (or false alarm) will be reported as the contact. ASSET only allows one detection (real or false) per MPA mission.

3. Evaluations of Current Model

[SH 91] has indicated several drawbacks in current ASSET's MPA model. Targets outside of the search region at the beginning of MPA's on station are excluded the search. So an MPA can not detect targets crossing the search region boundary during its searching.

The ability of an MPA to detect and kill a target is affected by the MPA and submarine false alarm rate. Apparently due to the limitation that engaged platforms are not detectable by opponent platforms, even extremely small submarine false alarm rates significantly reduce the detectability of the submarine.

In addition, currently only a single contact is performed per MPA mission, even if that contact is a false alarm and the MPA has sufficient time to

remain on-station. After the MPA completes one engagement, no further search is conducted.

B. ALLOCATION MODEL

In ASSET each antisubmarine warfare operational center (ASWOC) has a specific non-overlapping ocean area of responsibility and several assigned MPA squadrons, both defined by the user. Based upon fused submarine picture obtained from the ASW fusion cer. r, ASWOCs cue their MPA assets to investigate areas that are likely to contain targets of interest. Remaining MPAs also may be assigned to perform uncued area search. ASWOCs make their MPA search assignments decisions regularly at user-specified allocation intervals. At each allocation interval, the fusion center will provide the ASWOC with an 86% search probability area (SPA) for each suspected target within the ASWOC's ocean area.

ASSET prepares a table that matches all pairs of available MPA and SPAs, giving the times of station, transit time to the SPA, and SPA size at mid-time on-station. Available MPAs include all MPAs on the ground in a ready status and those MPAs conducting uncued search that have not yet reported a detection.

ASSET first eliminates each SPA/MPA pairs whose projected SPA size exceeds a user-entered maximum, and pairs whose computed time on-station is less than the user-entered minimum (uncued, divertable MPA may have a different user-entered minimum). From the remaining pairs, ASSET selects the first SPA/MPA combinations with minimum transit time. This process continues until either all the SPAs or all the MPA have been exhausted. If any available MPA remains, up to a user-entered maximum will be assigned to search a user-designated uncued search region. The MPA's search region is randomly located inside this designated uncued area search region.

[SH 91] points out that the way ASSET selects SPA/MPA pairs can be improved. Also ASSET only allocates MPA at the predetermined allocation intervals.

C. IMPROVEMENTS

[SH 91] has suggested several alternate models of MPA. Most of the MPA modeling improvements mentioned here are derived from that report but with some modifications.

1. Glimpse Rate Model

All the continuous detection sensors in ASSET except the MPA use a glimpse rate to determine detection opportunities. An MPA Glimpse Rate Model (MGRM) approximates a continuous-looking sensor pattern that has a probability of detection (P_d) of less than 1.0 with a glimpsing sensor region that has a P_d of 1.0 (as ASSET currently does with the tripwire sensor). The sensor region would then be glimpsed to provide a detection rate identical with that obtained by a continuous sensor conducting a random search.

Detections for MGRM are based on the random search model where the detection rate is the ratio of the relative area searched per unit time over the total area of search region. Random search predicts that a target, moving randomly through a field of continuous stationary sensors, will be detected at a certain rate. By glimpsing the sensor field at this detection rate, MGRM can produce an identical detection rate regardless of how the target is moving.

a. Area of the Search Region

For a cued search, the sonobuoys will be uniformly placed through the circular search region of tracker-correlator's best estimate of target position.

Otherwise, the search region will be randomly placed within the user-entered uncued search region.

b. Determination of Detection

When an MPA arrives on-station, MGRM will calculate a glimpse rate and initial glimpse time for each potentially detectable target. As in ASSET's other glimpse models, a target is deemed potentially detectable if its range from the MPA is less than its maximum speed multiplied by the MPA search time. The initial glimpse time will be determined from an exponential random draw using the inverse of the glimpse rate (or detection rate) as the mean glimpse interval. Detection rate will be computed as ASSET currently does. If the target is within the SPA when the glimpse occurs it is considered a successful detection.

c. Effect of False Contacts

The false contacts are generated by a Poisson process with a user-entered false contact rate. An exponentially distributed random number will be drawn to determine the time of the false contact. It will transmit a false target report at the time of false alert if the MPA is still on station and has not yet been killed. The MPA will spend the same amount of buoys for target classification, as it does in a normal engagement. A false contact will not be processed if the MPA is engaged at the time of the false alert.

2. MPA Allocation Model

a. Determination of SPA/MPA Matching

The allocation of the next MPA would be to the MPA/SPA pair with the largest ratio of MPA's time on station to SPA size. This will increase cumulative detection probability without increasing the processing time to perform an allocation.

b. After Prosecuting a Target

When the MPA finishes the prosecution of a submarine, it will stay in the search region until the end of its time on station. This will enable the MPA to have opportunities to detect other targets. Also if time permits, the MPA can be directed to other cues.

D. IMPLEMENTATION DETAILS

This section provides implementation details for the suggested improvements. They are categorized by their physical locations in the application package distributed by Metron.

1. Structure File

a. MPAstruct

The MPAstruct defines the operation characteristics of an MPA. One data field is added:

`engagedP`	Indicates whether an MPA is currently engaging with opponent platform

2. Class ASWOC

The ASWOC class describes the ASWOC's geographical location, assigned squadrons, specific ocean area of responsibility, and operations. We redefine two of its methods:

a. Modified Functions

`makeMPAassignment`: The selection of next cue is the cue with the largest ratio of MPA's time on station over SPA size.

`getSearchValues`: SearchValues now contains searchTime, missionCount, MPA, SPA size.

3. Class MPAsquadron

The class MPAsquadron defines variables of its command, operation characteristics, weapon status, and operations. We describe the modification in following sections:

a. Modified Functions

`dispatchMPAtoAreaSearch & dispatchMPAtoCue:` Schedule detection opportunities for the MPA to every possible submarine target based on the inversion of detection rate to the target.

`endMPAflight:` Resets MPAstruct-engagedp flag to NIL after finishing its flight.

`beginSearch:` Schedules detection opportunity for the MPA to every possible submarine target based on the inversion of detection rate to the target.

`localizeTarget:` Checks if the MPA is on station and the target is active before doing anything.

`loseMPA:` Checks if the MPA has not being killed yet before doing anything.

b. New Functions

`setNextDetectionOpportunity:` Schedules next detection opportunity for MPA.

`processDetectionOpportunity:` Determines target detection; schedule next detection opportunity.

`targetDetectedp:` Determines it the target is detected by checking if the target is inside search region.

targetCoveredp: Checks if the submarine is inside the given search region.

setNextFalseAlarm: Schedules false alarm event during MPA's time on station by making exponential draw from mean time between false alarm.

processFalseAlarm: Transmit a false target report if the MPA is still on station and not being killed; expends buoys for target classification; schedule next false alarm event.

endEngagement: Reset MPAstruct-engagedp flag to Nil.

transmitFalseAlarmReport: Transmit the false target report to its command.

prosecuteTarget: Determines who wins during engagement; if time permit and still carry enough weapons MPA will stay for another detection opportunity.

c. Deleted Functions

getSubInSearchArea

V. CONVERTING ASSET TO CLOS

The ASSET was implemented in Macintosh Allegro Common Lisp Version 1.3.2 which uses Object Lisp as object-oriented extension of Common Lisp. Recently Apple's support of this version was stopped, and CLOS became the standardized Common Lisp package for version 2.0 release. For the benefit of future maintenance of the ASSET, we decide to convert the current code to CLOS.

This chapter provides a summary of the differences between Object Lisp and CLOS, the work of converting current ASSET code to CLOS, and the results we have achieved.

A. DIFFERENCES BETWEEN CLOS AND OBJECT LISP

Machintosh Common Lisp (MCL) Version 1.3.2 uses the object protocol Object Lisp which supports multiple inheritance but only simple method combination. In Object Lisp one could make an object submarine with an instance SSN 688, then create SSN 977 which inherits from SSN 688, and SSN 755 which inherits form SSN 977. CLOS uses a class-instance protocol. In CLOS we cannot make a subclass from SSN 688, since it is not a class. Instead, we must create another class, possibly a subclass of submarine.

Binding and scoping have changed substantially. Instead of object variables, there are slots in CLOS classes and instances. Rather than asking an object to run its version of an object function, CLOS applies methods of generic functions to instances. The detailed information of difference between Object Lisp and CLOS

can be found in the user's manual of Macintosh Allegro Common Lisp Version 1.3.2 and [ST 90].

B. CONVERTING TO CLOS

1. Class Definition

Generally in MCL 1.3, `defobject` defines the class hierarchy, and `exist` creates an object instance and sets its own binding of variables to initial values by using function `have`. Most frequently, the corresponding material in CLOS goes into the `defclass` with its slot specifiers. The `initialize-instance` specifies the values of slots that cannot initialized with initialization arguments or initial forms. It also perform any other necessary initialization. If `have` is used dynamically in the program, one can transform uses of `have` into static slots and values in `defclass`.

2. Method Definition

All object functions must be turned into generic functions. That is, every piece of code of the form

```
(defobfun (Die submarine) arguments body)
```

must be transformed to

```
(defmethod Die ((submarine submarine) &rest-of-args)
    body)
```

3. Replace Free Variables References

Since the slot-values and variables do not access the same namespace, we must bring all free variable references into the slot namespace. We have two ways of dealing with free variables: set them explicitly with `slot-value` or use accessor calls. Using accessors are usually preferable, since they allow us to change the representation of our classes without changing any of the user code.

4. **Remove Calls to Ask**

Instead of `ask`, we now use `slot-value` or an accessor for a value, or a call to a method. Depending on the context, we should choose the appropriate call. We must run methods on instances of a class. All calls must now be directed to instances, not to class object. An Object Lisp asking for the object function setForStart

```
(ask submarine (setForStart))
```

becomes a call to the appropriate method of setForStart for an instance of acousticPlatform, SSN688:

```
(setf SSN688 (make-instance 'submarine))
(setForStart SSN688)
```

5. **Create Instance**

Rather than creating all new objects with `oneof`, we create instances with `make-instance`. Similarly, we replace `kindof` with `defclass`. Note that in CLOS, we cannot make instances directly into classes. Instead, we make new classes based on the class we want to specialize, then create instances of those classes.

6. **Change Function Names**

Numerous functions names and other symbols have changed in both Common Lisp and Macintosh Common Lisp. Functions involving dialog items or windows are frequently affected.

7. **Changes of Dialogs**

The implementation of dialog has changed substantially in MCL Version 2.0. Dialogs as a separate class have disappeared. Dialog items may now be

added to all views. Some functions have changed to reflect the new definition of dialog.

C. RESULTS

The source code of ASSET contains about 29,000 lines. Most the work of converting it to CLOS can be done by mechanically transforming the source code. We used a conversion utility program to help converting the class and method definitions, referencing a free variable, running a method, creating a instance, and changing function names. This utility program will accept a text file (of MACL) and try to convert it to MCL program. Unfortunately, due to time constraints we are not able to make the utility program a total solution for the porting. To complete the conversion, we still need to manually convert all window system to the newer version.

VI. CONCLUSION

A. SUMMARY

In this thesis, we presented the idea of object-oriented simulation. In addition, we implemented the improvements suggested in previous evaluations of ASSET in NPS. Some modifications to those suggestions were made during the implementation process. Part of the current ASSET source code is converted to CLOS but more work is needed to complete the task.

1. Detection Model

A compound Lambda-Sigma jump detection model was implemented to simulate the detection of submarines. Also the target's most detectable frequency and detection rate were allowed to vary with environmental region. Multiple engagements between platforms were allowed.

2. MPA Model

A glimpse rate model was used to determine detection opportunities of MPA and to approximate a continuous-looking sensor pattern. The glimpsing sensor field detects a target which is within the sensor region at the time of a glimpse with a probability of detection of 1.0.

ASSET allocates MPA to cues generated from the tracker-correlator by selecting SPA/MPA pairs using the ratio of MPA's time on-station to the SPA size as the selection criterion. If time permits, the MPA will stay on search region after prosecuting a submarine for another detection opportunity.

3. Converting ASSET code to CLOS

We investigated the feasibility and technique of converting current ASSET code to CLOS. We used a conversion utility to help us transform the current ASSET code to CLOS. Due to time constraints, only part of the converting task was completed.

B. FUTURE WORK

There are several areas that could be enhanced to improve the ASSET. The user interface could be improved for easier setup of a simulation architecture. It is possible to speed up the overall performance of the ASSET by porting it to a SUN workstation with a more powerful CPU. Various modules such as surface ASW platforms, active and nonacoustic sensors could be added to form a more complete ASW simulation. In addition, it would be useful to add machine intelligence features into platform tactics to better represent smart platforms. The bottle neck of ASSET performance has not been exactly located. It is possible to further speed up ASSET's performance.

APPENDIX A MODIFIED MODULES OF ASSET CODE

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; CHANGE LOG:
;;
;; in (defstruct platformAcousticParms)
;; new data field added for individual lambda-sigma process and
;; detection model:
;;  (individualSigma 0)          sigma for individual fluctuation
;;  (lambdaSigmaValue 0)                fluctuation of target's signal excess
;;  (meanInterval 1000)          mean time between fluctuation
;;  (detectableInterval 1)       target's detectabl interval(1/lambda)
;;                               for scheduling detection event
;;  (env-freq-list nil)          environment-frequency pairs list for a
;;                               target used to determine its radiated
;;                               frequjency in a particular environment
;;
;; in (defstruct MPAstruct)
;; new data field added for MPA's detection model:
;; indicates whether a MPA is engaging with (or been engaged by) a
;; target
;;  (engagedp        nil)
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

  (defstruct coast-event
     time
     object
     procedure
     data
     updateList
  )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

  (defstruct coast-message
     send-time             ; Time message sent.
     receipt-time          ; Time message received at last node.
     type                   ; E.g. 'Detect-msg
   sender
     content
     size
```

```
    transmission-path
    transmission-count
  )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

  (defstruct obu-contact
    id
    receipt-time
    track-association
    sensor
    categorization
    spatial-data
    altitude
    HFDFp
  )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

  (defstruct obu-track
    id
    number-of-contacts
    head-state-covariance
    head-contact-id
    head-spatial-data
    tail-state-covariance
    tail-contact-id
    tail-spatial-data
    altitude
    (contacts nil :type list)
  )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

  (defstruct obu-data-field
    type
    obs-time
    lat
    lng
    brg
    maj
    min
    cse
```

```
      cse-unc
      spd
      spd-unc
   )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

   (defstruct obu-state-field
      ref-lat
      ref-lng
      (state      nil :type (array float    4  ))
      (covariance nil :type (array float '(4 4)))
   )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

   (defstruct obu-sensor
      name
      single-report-to-trackp
   )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

   (defstruct obu-category
      pinnedp
      lockedp
      assoc-count
      cluster-count
      ambiguity-list
      ambiguity-resolution
      ambiguity-resolution-time
   )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

  (defstruct sen-detect-parms
     sensorName
     ga
     bw
     pd
     pfa
     rho
     msize
```

```
    min-delay
    max-delay
    type
    pos-unc
    spd-unc
    cse-unc
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defstruct target-state-parms
    targetName
    lat
    lng
    cse
    spd
    detectablep
    communicatingp
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defstruct platformStateParms
    (platformName           "")
    (platformType       'general)
    (classType          nil)
    (generalCommType    'surface)
    (targetType         nil)
    (side               nil)
    (opponents          nil)
    (lat                0)
    (lng                0)
    (hgt                0)
    (cse                0)
    (spd                0)
    (nav                0)
    (activep            t)
    (communicatingp     nil)
    (engagedp           nil)
    (engagementInterval nil)
    (engagedPlatform    nil)
    (inTrailp           nil)
    (trailedPlatform    nil)
```

```
    (cuedp               nil)
    (cueingInterval        nil)
    (interceptingp         nil)
    (targetObject          nil)
    (postEngageMotionOrder   nil)
    (deletedp              nil)

  )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

  (defstruct platformAcousticParms
    (sigFreq         nil)
    (spdThresh         0)
    (RNslow           0)
    (RNfast          0)
    (SNlist        nil)
    (DI-RD-list   nil)
    (sigFreq     nil)
    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    ;;
    ;; new data field added for individual lambda-sigma
    ;; process
    ;;
    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    (individualSigma 0)
    (lambdaSigmaValue 0)       ;fluctuation of target's signal excess
    (meanInterval 1000)                ;mean time between fluctuation
    (detectableInterval 1)     ;target's detectabl interval(1/lambda)
                               ;for scheduling detection event
    (env-freq-list   nil)      ;environment-frequency pairs list for a
                               ;target;  used to determine its radiated
                               ;frequjency in a particular environment
  )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

  (defstruct locationOrder
    lat
    lng
    hgt
  )
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defstruct movementOrder
    time
    cse
    tLat
    tLng
    spd
    nav
    area
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defstruct patrolOrder
    time
    region
    patrolDuration
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defstruct transitOrder
    time
    track
    legnumber
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defstruct planOrder
    time
    plan
    stepnumber
    duration
    legnumber
    repositionp
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defstruct region-parms
    regionName
```

```
     cPoints
     minMax
     minLat
     maxLat
     minLng
     maxLng
     cenLat
     cenLng
     gridpts
     ptArea
     totalArea
     checkpts
     thePoles
     theNormals
     theRngs
     theBrgs
     cVectors
     sVectors )
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

  (defstruct trackParms
     endPoints
     legTotal
  )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

  (defstruct planParms
     planlist
     stepTotal
  )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

  (defstruct coverage-parms
     chkLng
     range
     time
  )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

  (defstruct posit
```

```
    lat
    lng
    hgt
    time
  )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

  (defstruct MPAstruct
    ASWOC
    squadron
    baseLat
    baseLng
    (readyp             t)
    (searchingp         nil)
    searchStartTime
    searchEndTime
    missionEndTime
    lat
    lng
    radius
    searchRegion
    (buoyCount          0)
    (torpedoCount       0)
    (totalFlightHours   0)
    (failureTime        0)
    (missionCount       0)

    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    ;;  new data field added for MPA's detection model;
    ;;  indicates whether a MPA is engaging with(or been engaged
    ;;  by) a target
    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    (engagedp           nil)
    drawnMission
    keepOutOrder
    nextContact
    kiiledp
  )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

  (defstruct spaceSensorStruct
```

```
  key1
  key2
  booleanKeys
  key1bins
  key2bins
  velocityBins
  depthBins
  probabilities
 )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defstruct sensorPd
  FAR
  WirePd
  buoyPdVec
  boatPdArray
 )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defstruct NAArea
  NA-index
  windSpeed
  windDirection
  visibility
  surfAirTemp
  seaSurfTemp
  cloudCover
  JerlovType
  bioluminesence
 )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defstruct forceStruct
  side
  sideColor
  menuTitle
  forceDefinition
  workSpaceList
  workSpaceFile
  classFile
```

```
 noiseFrequencyFile
 noiseFrequencies
 objectKeyList
 instanceList
 opponentList
 linkList
 connectList
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defStruct objectStruct
 template
 index
 (classNameList  ())
 (classList      ())
 (objectNameList ())
 (objectList     ())
 menuTitle
 disabledp
 dialogText
 classObject
 platformType
 initList
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defstruct subobject-category
 name
 parent-subobject
 descendent-subobject-list
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defstruct radio-button
 symbol
 value
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

;; (4) COAST MOE CALCULATION STRUCTURES:

```
(defstruct coast-moe
   (number-of-data-elements        0)
   (sum-of-data                    0)
   (sum-of-squared-data      0)
   (data                     nil)
   (abscissa                 0)
)
```

;;

```
(defstruct MOEstruct

  menuTitle
  headings
  CGheadings
  side
  eventType
  initialcount
  count
  MOEarray
  currentIndex
  elapsedTime
)
```

;;

```
(defstruct MOEeventStruct
  repNumber
  time
  eventID
  object1
  object2
  data
)
```

;;

```
(defstruct MOEobjectStruct
  side
  type
  class
```

```
  name
)
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; CHANGE LOG:
;;
;; lambdaSigmaProcess-rb added
;;   radio button for inputting lamda-sigma parameters
;;
;; env-freq-rb added
;;   radion button for inputting environment-frequency pairs of
;;   a acousticplatform
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;..... LOAD "Object Editor RBs" Folder Contents:

(let ((directory "coastLibrary;FolderH - RBs:")
     (files    (list "general RB type"
                     "data-edit-rb"
                     "Lat Lng RB"
                     "Select and Modify RB"
                     "Table Selector RB"
                     "Build Connectivities RB"))
   )
   ( dolist (i files)
         (load (make-pathname :directory directory :name i))
   )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(ask objectEditor

   (have 'rb-installation-list nil)

   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

   (have 'Sensor-rb
      (oneof select-&-modify-rb
            :dialog-item-text "Sensors"
            :window-title "Select Sensors"
            :dialog-item-position (make-point -1 391)
            :dialog-item-size (make-point 140 16)
            :radio-button-pushed-p nil
```

```
        :radio-button-cluster '0
        :rb-symbol 'Sensor-rb)
)

(have 'Link-rb
   (oneof select-&-modify-rb
        :dialog-item-text "C3 Links"
        :window-title "Select Links"
        :dialog-item-position (make-point -1 408)
        :dialog-item-size (make-point 140 16)
        :radio-button-pushed-p nil
        :radio-button-cluster '0
        :rb-symbol 'Link-rb)
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(have 'Connect-rb
  (oneof ConnectivityRadioButton
       :dialog-item-text "C3 Connectivities"
       :dialog-item-position (make-point -1 426)
       :dialog-item-size (make-point 140 16)
       :radio-button-pushed-p nil
       :radio-button-cluster '0
       :rb-symbol 'Connect-rb)
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(have 'Location-rb
  (oneof lat-lon-rb
       :dialog-item-text "Location"
       :dialog-item-position (make-point -1 338)
       :dialog-item-size (make-point 140 16)
       :radio-button-pushed-p nil
       :radio-button-cluster '0
       :rb-symbol 'Location-rb)
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(have 'Region-rb
   (oneof table-selector-rb
```

```
        :dialog-item-text "Region"
        :dialog-item-position (make-point -1 374)
        :dialog-item-size (make-point 140 16)
        :radio-button-pushed-p nil
        :radio-button-cluster '0
        :rb-symbol 'Region-rb
        :table-data-form '(ask regionManager rgnList))
)

(have 'Tripwire-rb
   (oneof table-selector-rb
        :dialog-item-text "Tripwire"
        :dialog-item-position (make-point -1 374)
        :dialog-item-size (make-point 140 16)
        :radio-button-pushed-p nil
        :radio-button-cluster '0
        :rb-symbol 'Tripwire-rb
        :table-data-form '(ask trackManager trackList))
)

(have 'Sensor-Region-rb
   (oneof table-selector-rb
        :dialog-item-text "Sensor Region"
        :dialog-item-position (make-point -1 374)
        :dialog-item-size (make-point 140 16)
        :radio-button-pushed-p nil
        :radio-button-cluster '0
        :rb-symbol 'Sensor-Region-rb
        :table-data-form '(ask regionManager rgnList))
)

(have 'Operating-Region-rb
   (oneof table-selector-rb
        :dialog-item-text "Operating Region"
        :dialog-item-position (make-point -1 374)
        :dialog-item-size (make-point 140 16)
        :radio-button-pushed-p nil
        :radio-button-cluster '0
        :rb-symbol 'Operating-Region-rb
        :table-data-form '(ask regionManager rgnList))
)

(have 'Area-Search-Region-rb
```

```
   (oneof table-selector-rb
        :dialog-item-text "Area Search Region"
        :dialog-item-position (make-point -1 374)
        :dialog-item-size (make-point 140 16)
        :radio-button-pushed-p nil
        :radio-button-cluster '0
        :rb-symbol 'Area-Search-Region-rb
        :table-data-form '(ask regionManager rgnList))
)

(have 'Plan-rb
   (oneof table-selector-rb
        :dialog-item-text "Motion Plan"
        :dialog-item-position (make-point -1 374)
        :dialog-item-size (make-point 140 16)
        :radio-button-pushed-p nil
        :radio-button-cluster '0
        :rb-symbol 'Plan-rb
        :table-data-form '(ask PlanManager motionPlanList))
)

(have 'Replenishment-rb
   (oneof table-selector-rb
        :dialog-item-text "Replenishment Plan"
        :dialog-item-position (make-point -1 374)
        :dialog-item-size (make-point 140 16)
        :radio-button-pushed-p nil
        :radio-button-cluster '0
        :rb-symbol 'Replenishment-rb
        :table-data-form '(ask PlanManager motionPlanList))
)

(have 'commander-rb
   (oneof table-selector-rb
        :dialog-item-text "Commander"
        :dialog-item-position (make-point -1 374)
        :dialog-item-size (make-point 140 16)
        :radio-button-pushed-p nil
        :radio-button-cluster '0
        :rb-symbol 'commander-rb
        :table-data-form
        '(let ((side  (ask my-dialog
                    (ask currentObject
```

```
                    (platformStateParms-side platformState))))
               (indexList (list (ask my-dialog (ask currentObject
                                                    commandIndex)))))
          (getNamesFromIndices side indexList)))
)

(have 'fusionCenter-rb
   (oneof table-selector-rb
        :dialog-item-text "Fusion Center"
        :dialog-item-position (make-point -1 374)
        :dialog-item-size (make-point 140 16)
        :radio-button-pushed-p nil
        :radio-button-cluster '0
        :rb-symbol 'fusionCenter-rb
        :table-data-form
        '(let ((side  (ask my-dialog
                    (ask currentObject
                     (platformStateParms-side platformState))))
               (indexList (list (ask my-dialog (ask currentObject
                                                    fusionIndex)))))
          (getNamesFromIndices side indexList)))
)

(have 'frequency-rb
   (oneof table-selector-rb
        :dialog-item-text "Signature Freq"
        :dialog-item-position (make-point -1 374)
        :dialog-item-size (make-point 140 16)
        :radio-button-pushed-p nil
        :radio-button-cluster '0
        :rb-symbol 'frequency-rb
        :table-data-form
        '(ask my-dialog
          (ask currentObject
               (getFreqList (platformStateParms-side
                                    platformState)))))
)

(have 'subMissions-rb
   (oneof table-selector-rb
        :dialog-item-text "Select Mission"
        :dialog-item-position (make-point -1 374)
        :dialog-item-size (make-point 140 16)
```

```
        :radio-button-pushed-p nil
        :radio-button-cluster '0
        :rb-symbol 'subMissions-rb
        :table-data-form '(ask my-dialog (ask currentObject
                                                subMissionList)))
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(have 'AcousSig-rb
  (oneof data-edit-rb
        :dialog-item-text "Acoustic Sig"
        :dialog-item-position (make-point  -1 374)
        :dialog-item-size (make-point 140 16)
        :radio-button-pushed-p nil
        :radio-button-cluster '0
        :data-key :acousticSig
  )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; new radio button for acoustic platform
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(have 'LambdaSigmaProcess-rb
  (oneof data-edit-rb
        :dialog-item-text "Lambda-Sigma Process"
        :dialog-item-position (make-point  -1 374)
        :dialog-item-size (make-point 140 16)
        :radio-button-pushed-p nil
        :radio-button-cluster '0
        :data-key :LambdaSigmaProcess
  )
)


(have 'SubmarineWeapons1-rb
  (oneof data-edit-rb
        :dialog-item-text "Submarine Weapons 1"
        :dialog-item-position (make-point  -1 374)
        :dialog-item-size (make-point 140 16)
        :radio-button-pushed-p nil
```

```
        :radio-button-cluster '0
        :data-key :SubmarineWeapons1
  )
)

(have 'SubmarineWeapons2-rb
  (oneof data-edit-rb
        :dialog-item-text "Submarine Weapons 2"
        :dialog-item-position (make-point  -1 374)
        :dialog-item-size (make-point 140 16)
        :radio-button-pushed-p nil
        :radio-button-cluster '0
        :data-key :SubmarineWeapons2
  )
)

(have 'SubmarineTactics-rb
  (oneof data-edit-rb
        :dialog-item-text "Submarine Tactics"
        :dialog-item-position (make-point  -1 374)
        :dialog-item-size (make-point 140 16)
        :radio-button-pushed-p nil
        :radio-button-cluster '0
        :data-key :SubmarineTactics
  )
)

(have 'sensorParms-rb
  (oneof data-edit-rb
        :dialog-item-text "Detection Parameters"
        :dialog-item-position (make-point  -1 374)
        :dialog-item-size (make-point 140 16)
        :radio-button-pushed-p nil
        :radio-button-cluster '0
        :data-key :sensorParms
  )
)

(have 'MPASquadron-rb
  (oneof data-edit-rb
        :dialog-item-text "Squadron Parameters"
        :dialog-item-position (make-point  -1 374)
        :dialog-item-size (make-point 140 16)
```

```
        :radio-button-pushed-p nil
        :radio-button-cluster '0
        :data-key :MPASquadron
  )
)

(have 'MPAstores-rb
  (oneof data-edit-rb
        :dialog-item-text "Stores Parameters"
        :dialog-item-position (make-point  -1 374)
        :dialog-item-size (make-point 140 16)
        :radio-button-pushed-p nil
        :radio-button-cluster '0
        :data-key :MPAstores
  )
)


(have 'general-motion-rb
  (oneof data-edit-rb
        :dialog-item-text "General Motion"
        :dialog-item-position (make-point  -1 374)
        :dialog-item-size (make-point 140 16)
        :radio-button-pushed-p nil
        :radio-button-cluster '0
        :data-key :generalMotion
  )
)

(have 'reports-rb
  (oneof data-edit-rb
        :dialog-item-text "Detection Reports"
        :dialog-item-position (make-point  -1 374)
        :dialog-item-size (make-point 140 16)
        :radio-button-pushed-p nil
        :radio-button-cluster '0
        :data-key :reports
  )
)

(have 'orbitParms-rb
  (oneof data-edit-rb
        :dialog-item-text "Orbital Parameters"
```

```
        :dialog-item-position (make-point  -1 374)
        :dialog-item-size (make-point 140 16)
        :radio-button-pushed-p nil
        :radio-button-cluster '0
        :data-key :orbitParms
  )
)

(have 'SOAallocation-rb
  (oneof data-edit-rb
        :dialog-item-text "Alloc Parameters"
        :dialog-item-position (make-point  -1 374)
        :dialog-item-size (make-point 140 16)
        :radio-button-pushed-p nil
        :radio-button-cluster '0
        :data-key :SOAallocationParms
  )
)

(have 'ASWOCallocation-rb
  (oneof data-edit-rb
        :dialog-item-text "Alloc Parameters"
        :dialog-item-position (make-point  -1 374)
        :dialog-item-size (make-point 140 16)
        :radio-button-pushed-p nil
        :radio-button-cluster '0
        :data-key :ASWOCallocationParms
  )
)

(have 'HFDF-rb
  (oneof data-edit-rb
        :dialog-item-text "HFDF Detection"
        :dialog-item-position (make-point  -1 374)
        :dialog-item-size (make-point 140 16)
        :radio-button-pushed-p nil
        :radio-button-cluster '0
        :data-key :HFDFparms
  )
)

(have 'minefield-rb
  (oneof data-edit-rb
```

```
        :dialog-item-text "Minefield Parameters"
        :dialog-item-position (make-point  -1 374)
        :dialog-item-size (make-point 140 16)
        :radio-button-pushed-p nil
        :radio-button-cluster '0
        :data-key :MineFieldParms
  )
)

(have 'tripwireParms-rb
 (oneof data-edit-rb
        :dialog-item-text "Tripwire Parameters"
        :dialog-item-position (make-point  -1 374)
        :dialog-item-size (make-point 140 16)
        :radio-button-pushed-p nil
        :radio-button-cluster '0
        :data-key :TripwireParms
  )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(have 'submarineSensorPD-rb
 (oneof coast-RB-type
        :dialog-item-text "Sub Sensor PDs"
        :dialog-item-position (make-point  -1 374)
        :dialog-item-size (make-point 140 16)
        :radio-button-pushed-p nil
        :radio-button-cluster '0
        :dialog-item-action
        '(ask (ask my-dialog currentObject) (editSubmarinePDs))
  )
)

(have 'surfaceSensorPD-rb
 (oneof coast-RB-type
        :dialog-item-text "Surf Sensor PDs"
        :dialog-item-position (make-point  -1 374)
        :dialog-item-size (make-point 140 16)
        :radio-button-pushed-p nil
        :radio-button-cluster '0
        :dialog-item-action
        '(ask (ask my-dialog currentObject) (editSurfacePDs))
```

```
  )
 )

 (have 'subSubPk-rb
  (oneof coast-RB-type
        :dialog-item-text "Sub Sub PKs"
        :dialog-item-position (make-point -1 374)
        :dialog-item-size (make-point 140 16)
        :radio-button-pushed-p nil
        :radio-button-cluster '0
        :dialog-item-action
        '(ask (ask my-dialog currentObject) (editSubSubPks))
  )
 )

 (have 'MPASubPk-rb
  (oneof coast-RB-type
        :dialog-item-text "MPA Sub PKs"
        :dialog-item-position (make-point -1 374)
        :dialog-item-size (make-point 140 16)
        :radio-button-pushed-p nil
        :radio-button-cluster '0
        :dialog-item-action
        '(ask (ask my-dialog currentObject) (editMPASubPks))
  )
 )

 (have 'minefieldRange-rb
  (oneof coast-RB-type
        :dialog-item-text "Sub Mine Ranges"
        :dialog-item-position (make-point -1 374)
        :dialog-item-size (make-point 140 16)
        :radio-button-pushed-p nil
        :radio-button-cluster '0
        :dialog-item-action
        '(ask (ask my-dialog currentObject) (editMineFieldRanges))
  )
 )

 (have 'minefieldPk-rb
  (oneof coast-RB-type
        :dialog-item-text "Minefield Pks"
        :dialog-item-position (make-point -1 374)
```

```
        :dialog-item-size (make-point 140 16)
        :radio-button-pushed-p nil
        :radio-button-cluster '0
        :dialog-item-action
        '(ask (ask my-dialog currentObject) (editMinePks))
  )
)

(have 'SN-rb
 (oneof coast-RB-type
        :dialog-item-text "Self Noise"
        :dialog-item-position (make-point -1 374)
        :dialog-item-size (make-point 140 16)
        :radio-button-pushed-p nil
        :radio-button-cluster '0
        :dialog-item-action
        '(ask (ask my-dialog currentObject) (editSN))
  )
)
(have 'DI-RD-rb
 (oneof coast-RB-type
        :dialog-item-text "DI, RD"
        :dialog-item-position (make-point -1 374)
        :dialog-item-size (make-point 140 16)
        :radio-button-pushed-p nil
        :radio-button-cluster '0
        :dialog-item-action
        '(ask (ask my-dialog currentObject) (editDI-RD))
  )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; new radio button for acoustic platform
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(have 'ENV-FREQ-rb
 (oneof coast-RB-type
        :dialog-item-text "Freq. - Environment"
        :dialog-item-position (make-point -1 374)
        :dialog-item-size (make-point 140 16)
        :radio-button-pushed-p nil
        :radio-button-cluster '0
```

```
            :dialog-item-action
             '(ask (ask my-dialog currentObject) (editEnv-Freq))
        )
      )

    )
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; CHANGE LOG:
;;
;;
;; data field "sigma" in EXIST function deleted;
;;
;; getPL-AN-Sigma modified
;;   target's radiated frequency is used on deciding PL & AN
;;
;; getSweepWidth modified
;;   target's radiated frequency is used on deciding PL & AN
;;   which in turn decide sweep width
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(setq environmentManager (kindof nil))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (exist environmentManager)(init-list)
  (usual-exist init-list)
  (have 'allPLfrequencies   nil)
  (have 'EnvRegionList              nil) ;form is: ({ Env-rgn obj1 } { Env-rgn
                                    ;obj2 } ...)
  (have 'PL-list                    nil)

  ;PL-list will be in following form:
  ; (Freq1 (PL-11 PL-12 ...) Freq2 (PL-21 PL-22 ...) . . .)
  ; where PL-ij is in form: ("PLname" "PLprimaryRgn" PL-array
  ; FreqAtWhichPLdefined)

  ; NOTE: FreqAtWhichPLdefined will equal FreqK in PL-list
  ;         immediately preceeding
  ;         the proploss list (ie all PL-ij's in (getf PL-list FreqK) will
  ;          be: ("PLname" "PLprimaryRgn" PL-array FreqK) ).
  ;         This info s redundant, but necessary since the proploss,
  ;         PL-ij, can be assigned to an env rgn at *any* frequency.
  ;         Thus, in order to re-constitute the proploss we need to
  ;         know the frequency at which it was defined (not the
  ;         frequency at which it is being used).

  (have 'defaultAN                      60.0)
```

```
  (have 'defaultInitialSphericalPL          66.2)
  (have 'defaultSpherical                   20.0)
  (have 'defaultInitialCylindricalPL        31.1)
  (have 'defaultCylindrical                 10.0)
  (have 'envFile (make-pathname
                   :directory "coastData;Environment Folder:"
                   :name      "EnvironmentFile"))

  (printStatus "LOADING ENVIRONMENTAL DATA" 2)
  (readEnvManager)

)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (writeEnvManager environmentManager) ()
  (with-open-file (toStream envFile
                    :direction :output
                    :if-exists :supersede
                    :if-does-not-exist :create
                  )
    (fresh-line toStream)
    (write allPLfrequencies :stream toStream)
    (terpri toStream)
    (write defaultAN :stream toStream)
    (terpri toStream)
    (write defaultInitialSphericaPL :stream toStream)
    (terpri toStream)
    (write defaultSpherical :stream toStream)
    (terpri toStream)
    (write defaultCylindrical :stream toStream)
    (terpri toStream)
    (write sigma :stream toStream)
    (terpri toStream)
    (write PL-list :stream toStream :escape t
          :level nil :length nil :array t)
    (terpri toStream)
    ;Save # of Env Rgns in file:
    (write (list-length EnvRegionList) :stream toStream)
    (dolist (thisEnvRgn EnvRegionList)
      (ask thisEnvRgn (writeEnvRgn toStream))
    )
    (terpri toStream)))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (readEnvManager environmentManager) ()
 (unless (probe-file envFile)
   (print "Warning from (readEnvManager environmentManager); file
                                                          does not exist:")
   (print envFile)
   (return-from readEnvManager nil)
 )
 (with-open-file (fromStream envFile
                        :direction :input
                        :if-does-not-exist nil
              )
   (let ((flength (file-length fromStream))
       (envManager (self))
       numberOfRgns thisRgn
       )
    (if (or (null flength) (< flength 5))
      (return-from readEnvManager nil)
    )
    (setq allPLfrequencies   (read  fromStream nil nil))
    (setq defaultAN          (read  fromStream nil nil))
    (setq defaultInitialSphericaPL  (read  fromStream nil nil))
    (setq defaultSpherical  (read  fromStream nil nil))
    (setq defaultCylindrical (read  fromStream nil nil))
    (setq sigma             (read  fromStream nil nil))
    (setq PL-list           (read  fromStream nil nil))
    (setq numberOfRgns      (read  fromStream nil nil))
    (setq EnvRegionList nil)
    (dotimes (k numberOfRgns)
     (setq thisRgn (oneof EnvRegionType))
     (ask thisRgn  (readEnvRgn fromStream envManager))
     (setq EnvRegionList (cons thisRgn EnvRegionList))
    )
    (setq EnvRegionList (reverse EnvRegionList))
   )
 )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; modified function
```

```
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (getPL-AN-Sigma environmentManager)(searcherObj targetObj)
 (let* ((sPos (ask searcherObj
           (list (platformStateParms-lat platformState)
                 (platformStateParms-lng platformState))))
     (tPos (ask targetObj
           (list (platformStateParms-lat platformState)
                 (platformStateParms-lng platformState))))
     (rng  (car (getRngBrg (car sPos)(cadr sPos)
                  (car tPos)(cadr tPos) *greatCircle*)))

     ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
     ;;
     ;; new local data used to obtain target's most
     ;; detectable radiated frequency
     ;;
     ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
     (targetEnvRgn          nil)
     (tagetEnvRgnName       nil)
     (searcherEnvRgn        nil)
     (ambNoise              nil)
     (PL-val                    nil)
     (AN-PL-pair            nil) freq  ;CHANGE
    )

  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
  ;;
  ;; this block modified to incorporat target's radiated frequency
  ;; on deciding PL & AN
  ;;
  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

  (dolist (thisEnvRgn EnvRegionList)
   (when (objectInRegionp searcherObj (ask thisEnvRgn geoRegion))
    (setq searcherEnvRgn thisEnvRgn)
    (return)
   )
  )
  (when searcherEnvRgn
   (dolist (thisEnvRgn EnvRegionList)
    (when (objectInRegionp targetobj (ask thisEnvRgn geoRegion))
```

```
        (setq targetEnvRgn thisEnvRgn)
        (return)
        ))
      (setq targetEnvRgnName (ask targetEnvRgn envRgnName))
      (setq freq (first (getf (ask targetobj
        (platformAcousticParms-env-freq-list
                    platformAcoustics)) targetEnvRgnName)))
      (setq AN-PL-pair (ask searcherEnvRgn (getf ambNoise&PropLoss
                                                    freq)))
      (setq ambNoise (first AN-PL-pair))
      (if (setq PL-array (third (second AN-PL-pair)))
        (setq PL-val (aref PL-array (round rng)))
      )
    )
    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    (if (null ambNoise) (setq ambNoise defaultAN))
    (if (null PL-val)   (setq PL-val (getDefaultSpherical rng)))
    (return-from getPL-AN-Sigma
          (values PL-val ambNoise sigma))
  )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; modified function
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (getSweepWidth environmentManager)(target NP)
  (let* (
        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        ;;
        ;; new local data used to obtain target's most
        ;; detectable radiated frequency
        ;;
        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        (targetEnvRgn nil)
        (tagetEnvRgnName nil)
        (freq  nil)

        (AN-PL-pair nil)
        (ambNoise nil)
```

```
    (PL-array nil)
   (FOM nil)
   (detectionRange 0)
   )
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; this block modified to incorporat target's
;; radiated frequency on deciding sweep width
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(dolist (thisEnvRgn EnvRegionList)
  (when (objectInRegionp target (ask thisEnvRgn geoRegion))
    (setq targetEnvRgn thisEnvRgn)
     (return)
  )
)
(when targetEnvRgn
  (setq targetEnvRgnName (ask targetEnvRgn envRgnName))
  (setq freq (first (getf (ask target
                 (platformAcousticParms-env-freq-list
                                platformAcoustics))
                                   targetEnvRgnName)))
  (setq AN-PL-pair (ask targetEnvRgn
                    (getf ambNoise&PropLoss freq)))
  (setq ambNoise (first AN-PL-pair))
   (setq PL-array (third (second AN-PL-pair)))
)
 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(if (null ambNoise) (setq ambNoise defaultAN))
(setq FOM (- NP ambNoise))
 (if (null PL-array)
    (setq detectionRange (RngFromPL&Spreading FOM 2))
     (dotimes (i (car (array-dimensions PL-array)))
       (if (aref PL-array i)
       (if (< (aref PL-array i) FOM)
        (setq detectionRange i))
       )
     )
)
 (return-from getSweepWidth
        (* 2 detectionRange))
)
```

```
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (getDefaultSpherical environmentManager)(rng)
  (if (< rng 0.001)(return-from getDefaultSpherical defaultInitialSphericalPL))
  (return-from getDefaultSpherical
      (+ defaultInitialSphericalPL (* defaultSpherical (log rng 10)))
  )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (RngFromPL&Spreading environmentManager) (PropLoss
SpreadExpt)
;;
;;    The spreading law exponent is 1 for cylindrical, 2 for Spherical
;;
;;    The PL=0 distance is assumed to be one yard, 1nm = 2027 yds.
;;
;;     Returns range in nm
;;
  (if (< PropLoss 0)
       (return-from RngFromPL&Spreading 0.001)
       (return-from RngFromPL&Spreading
               (float (/ (expt 10 (/ PropLoss (* 10 SpreadExpt)))
                    2027))
       )
  )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (getPLnames environmentManager) (freq)
  (let ((PLsAtFreq (getf PL-list freq "Not Found"))
     )
   (when (equalp PLsAtFreq "Not Found")
     (princ "Warning from getPLnames: no PLs at given freq ")
     (princ freq) (terpri)
     (return-from getPLnames nil)
   )
   (return-from getPLnames (mapcar #'first  PLsAtFreq))
```

```
  )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (getPLregions environmentManager) (freq)
  (let ((PLsAtFreq (getf PL-list freq "Not Found"))
        )
    (when (equalp PLsAtFreq "Not Found")
      (princ "Warning from getPLregions: no PLs at given freq ")
      (princ freq) (terpri)
      (return-from getPLregions nil)
    )
    (return-from getPLregions (mapcar #'second  PLsAtFreq))
  )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (getPLarray environmentManager) (freq PLname)
  (let ((PLsAtFreq (getf PL-list freq))
        )
    (dolist (PL PLsAtFreq)
      (if (string-equal (first PL) PLname)
        (return-from getPLarray (third PL)))
    )
    (princ "Warning from getPLarray: no PL with freq, name: ")
    (print (list freq PLname)) (terpri)
    (return-from getPLarray nil)
  )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (getPL environmentManager) (freq PLname)
  (let ((PLsAtFreq (getf PL-list freq))
        )
    (dolist (PL PLsAtFreq)
      (if (string-equal (first PL) PLname)
        (return-from getPL PL))
    )
    (princ "Warning from getPL: no PL with freq, name: ")
    (print (list freq PLname)) (terpri)
```

```
    (return-from getPL nil)
  )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (putPL environmentManager) (PL freq)
  (let ((PLsAtFreq (getf PL-list freq "Not Found"))
       )
    (when (equalp PLsAtFreq "Not Found")
      (princ "Warning from putPL: bad freq = ")
      (princ freq) (terpri)
      (return-from putPL nil)
    )
    (setf (getf PL-list freq) (cons PL PLsAtFreq))
  )
)
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; CHANGE LOG:
;;
;; new data fields and input dialogs of globallambda-sigma jump model
;; for signal excess calculation added; user is required to specify
;; them during data input stage:
;;  globalLambdaSigmaValue
;;  globalSigma
;;  globalMeanInterval
;;
;; setUpForReplication modified
;;  initializes globalLambdaSigmaValue and schedules event in
;;  exponentially separate time to update its value
;;
;; updateGlobalLambdaSigmaValue added to update global lambda
;; sigma value in exponentially separate time
;;
;; FOMglimpseInterval and its input dialog deleted;
;;  now using target's DetectableInterval to schedule next detection
;;  event
;;
;; setNextDetectionOpportunity modified
;;  it now allows engaged target being detectable to searcher that is
;;  not engaged; also check range when scheduling next detection
;;  opportunity for engaged platforms
;;
;; processDetectionOpportunity modified
;;  engaged searchers searcher won't detect other targets;
;;  engaged targets is detectable to other searchers
;;
;; getDetectionOppResult modified
;;  determines target's most detectable radiated frequecy;
;;  using global and individual lambda-sigma-value to adjust
;;  MSE(mean signal excess)
;;
;; processPlatformKill modified
;;  check if it was killed before
;;
;; setNextFalseAlarm and processFalseAlarm modified
;;  check if the searcher was killed before doing anything
;;  it won't set the engaged flag of searcher; so it can deal
;;  with real target while it is processing FA
```

```
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(setq Umpire (oneof nil))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (exist Umpire) (init-list)
 (usual-exist init-list)

  (have 'prehostilitiesDuration     50)
  (have 'hostilitiesDuration                200)

  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
  (have 'globalSigma                        0)
  (have 'globalMeanInterval                 1000) ;mean time between change
                                    ;of lambda-sigma signal excess
  (have 'globalLambdaSigmaValue       0) ;lambda-sigma signal
                              ;excess
  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

  (have 'maxClosingSpeed                    50)
  (have 'maxSubDetectionRange               100)
  (have 'maxSurfaceDetectionRange           100)
  (have 'hostilitiesp                       nil)
  (have 'name               "Simulation Umpire ")
  (have 'btitle             "EDIT Umpire Parameters")
  (have 'etitle             "EDIT Umpire Parameters")
  (have 'dataNameList
        '(:main  (prehostilitiesDuration hostilitiesDuration
                  globalSigma globalMeanInterval)))
  (have 'dataValueList
        `(:main (,prehostilitiesDuration ,hostilitiesDuration
       ,globalSigma ,globalMeanInterval)))
  (have 'dataTypeList
        '(:main (data data data data)))
  (have 'dataTextList
        '(:main ("Prehostilities Duration" "Hostilities Duration"
                              "Global Sigma" "Global Mean Int")))
  (have 'dataTemplateText
        '(:main (("Hours"   "[0 - unlimited]")("Hours"   "[0 - unlimited]")
                ("decibles" "[0 - 20]") ("Hours (1/Lambda)"
                                          "[0.25 - unlimited]"))))
```

```
 (have 'dummyRBdata             nil)
 (have 'Rb_datanamelist   nil)
 (have 'Rb_datavaluelist   nil)
 (have 'Rb_designatorlist  nil)
 (have 'UmpireDatafile
      (make-pathname
       :directory "coastData;Umpire Folder:"
               :name "Umpire Data File"))
 (printStatus "LOADING UMPIRE DATA" 2)
 (recallUmpireData)
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (editParameters Umpire) ()
 (let ((mySelf (self)))
    (editObject mySelf :main)
 )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (recallUmpireData Umpire) ()
 (with-open-file (rstream umpireDataFile
               :direction :input
               :if-does-not-exist :create)
   (setq dataValueList (read rstream nil dataValueList))
   (mapc #'(lambda(x y)
           (if (listp x)(mapc #'set x y)))
       dataNameList
       dataValueList)
 )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (saveUmpireData Umpire) ()
 (with-open-file (rstream umpireDataFile
                    :direction :output
                    :if-exists :supersede
                    :if-does-not-exist :create)
   (prin1 dataValueList rstream)
```

```
  )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; modified function
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (setUpForReplication Umpire) ()

  (let ((ctime (getCurrentTime))
       )
     (displayNotHostilitiesFlag)
     (setq hostilitiesp nil)

      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
      ;;
     ;; set up globalLambdaSigmaValue and schedule event in
     ;; exponentially separate time to update its value
      ;;
      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
     (setq globalLambdaSigmaValue (* globalSigma (normalDraw)))
     (addEvent (make-coast-event
         :time (+ cTime (exponentialDraw globalMeanInterval))
         :object (self)
         :procedure 'updateGlobalLambdaSigmaValue
         :data nil
         :updateList nil))
      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

     (beginFOMdetectionOpportunities cTime)
     (beginFalseAlarms cTime)
     (addEvent (make-coast-event
                 :time (+ prehostilitiesDuration cTime)
                 :object  (self)
                 :procedure 'processBeginHostilities
                 :data  nil
                 :updateList nil))
     (addEvent (make-coast-event
                 :time (+ prehostilitiesDuration hostilitiesDuration
                                                           cTime)
                 :object  (self)
```

```
                :procedure 'processEndOfReplication
                :data  nil
                :updateList nil))
  )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (processBeginHostilities Umpire) (cTime data)
  (setq hostilitiesp t)
  (displayHostilitiesFlag))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (processEndOfReplication Umpire) (cTime data)
  (setq hostilitiesp nil)
  (endReplication)
  (displayNotHostilitiesFlag)
  )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; new function added to update global lambda-sigma
;; value in exponentially separate time interval
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (updateGlobalLambdaSigmaValue Umpire) (cTime data)
  (setq globalLambdaSigmaValue (* globalSigma (normalDraw)))
  (addEvent (make-coast-event
            :time (+ cTime (exponentialDraw globalMeanInterval))
            :object (self)
            :procedure 'updateGlobalLambdaSigmaValue
            :data nil
            :updateList nil))
  )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (beginFOMdetectionOpportunities Umpire) (cTime)
  (dolist (sideStruct (ask objectManager forceStructures))
     (dolist (searcher (getf (forceStruct-instanceList sideStruct) :FOMsearchers))
       (dolist (target (getf (forceStruct-instanceList sideStruct) :FOMtargets))
```

```
        (setNextDetectionOpportunity ctime (list searcher target))))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; target's radiated frequency changes depending
;; which environment region the target is in;
;; if searcher is engaging the detection will be rescheduled with
;; exponential draw with target's detectable interval as mean
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (setNextDetectionOpportunity Umpire)(ctime platforms)
  (let* ((searcher      (car  platforms))
        (searcherState (ask searcher platformState))
        (target        (cadr platforms))
        (targetState   (ask target   platformState))
        (rng (car (getRngBrg (platformStateParms-lat searcherState)
                             (platformStateParms-lng searcherState)
                             (platformStateParms-lat targetState)
                             (platformStateParms-lng targetState)
                             *greatCircle*)))
       (maxDetectionRange
        (if (equalp 'submarineTarget
              (platformStateParms-targetType targetState))
                             maxSubDetectionRange
                             maxSurfaceDetectionRange))
       (separation (- rng maxDetectionRange))

       (targetAcoustics  (ask target   platformAcoustics))
       (targetEnvRgn nil)
       (tagetEnvRgnName nil)
       (envRgnList (ask environmentManager envRegionList))
       (SF nil) ;target's most detectable radiated frequency
       (meanDetectInterval
        (platformAcousticParms-detectableInterval
                             targetAcoustics))
    )
   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
   ;;
   ;; determine target's most detectable radiated frequecy
   ;;
   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
   (dolist (thisEnvRgn envRgnList)
     (when (ask environmentManager
```

```
        (objectInRegionp target (ask thisEnvRgn geoRegion)))
    (setq targetEnvRgn thisEnvRgn)
    (return)
    ))
(setq targetEnvRgnName (ask targetEnvRgn envRgnName))
(setq SF (first (getf (ask target
                        (platformAcousticParms-env freq-list
                                                   platformAcoustics))
                targetEnvRgnName)))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; check that both platforms are active
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    (when
     (and (platformStateParms-activep searcherState)
          (platformStateParms-activep  targetState))
     ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
     ;;
     ;; if searcher is engaged schedule next detection opportunity
     ;; based on earlist possible time these plafforms can come
     ;; within range or on mean detection interval whichever is
     ;; longer; now engaged target still detectable
     ;; by non-engaged searacher
     ;;
     ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
     (when (platformStateParms-engagedp        searcherState)
       (if (< 0 (- separation (* maxClosingSpeed
                                 meanDetectInterval)))
         (addEvent (make-coast-event
                    :time (+ cTime (max meanDetectInterval
                                        (/ separation maxClosingSpeed)))
                    :object  (self)
                    :procedure 'setNextDetectionOpportunity
                    :data  platforms
                    :updateList platforms)
           )
         (addEvent (make-coast-event
                    :time (+ cTime (exponentialDraw
                                      meanDetectInterval))
                    :object  (self)
                    :procedure 'setNextDetectionOpportunity
                    :data  platforms
                    :updateList platforms)
           )
```

```
        )
       (return-from setNextDetectionOpportunity nil)
       )
#|
  depending upon range: begin random glimpsing,
  or calculate earliest possible time these platforms can come within range
  (based on maxDectionRange MaxClosingSpeed and meanDetectInterval)
|#
       (if (< 0 (- separation (* maxClosingSpeed
                                  meanDetectInterval)))
          (addEvent (make-coast-event
                    :time (+ cTime (max meanDetectInterval
                                  (/ separation maxClosingSpeed)))
                    :object  (self)
                    :procedure 'setNextDetectionOpportunity
                    :data  platforms
                    :updateList platforms)
          )
          (addEvent (make-coast-event
                    :time (+ cTime (exponentialDraw
                                  meanDetectInterval))
                    :object  (self)
                    :procedure 'processDetectionOpportunity
                    :data  platforms
                    :updateList platforms)
          )
        )
       )
     )
 )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (processDetectionOpportunity Umpire)(ctime platforms)
  (let* ((searcher      (car  platforms))
       (target        (cadr platforms))
       (searcherState (ask searcher platformState))
       (targetState   (ask target   platformState))
       (maxDetectionRange
        (if (equalp 'submarineTarget
       (platformStateParms-targetType targetState))
          maxSubDetectionRange maxSurfaceDetectionRange))
```

```
    (targetAcoustics  (ask target  platformAcoustics))
    (meanDetectInterval
     (platformAcousticParms-detectableInterval
         targetAcoustics)) ;target's detetable interval
   )

  (when (and (platformStateParms-activep searcherState)
        (platformStateParms-activep  targetState))

   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
   ;;
   ;; engaged searcher searcher won't detect other targets
   ;; engaged target is detectable to other searchers
   ;;
   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
   (when (platformStateParms-engagedp        searcherState)
    (addEvent (make-coast-event
          :time (+ cTime (exponentialDraw meanDetectInterval))
          :object  (self)
          :procedure 'processDetectionOpportunity
          :data  platforms
          :updateList platforms)
         )
     (return-from processDetectionOpportunity nil)
    )

   (if (< (car (getRngBrg (platformStateParms-lat searcherState)
                 (platformStateParms-lng searcherState)
                 (platformStateParms-lat targetState)
                 (platformStateParms-lng targetState)
                 *greatCircle*))
      maxDetectionRange)
    (if (getDetectionOppResult searcher searcherState
                   target  targetState)
      (processTargetDetection ctime searcher searcherState
                   target  targetState))
    )
   (setNextDetectionOpportunity ctime platforms)
   )
  )
 )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;
;; modified function
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (getDetectionOppResult Umpire)(searcher searcherState target
targetState)
 (let ((searcherAcoustics (ask searcher platformAcoustics))
      (targetAcoustics  (ask target  platformAcoustics))
      (envRgnList (ask environmentManager envRegionList))
      (targetEnvRgn nil)
      (targetEnvRgnName nil)
      (SF nil)
      )
   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
   ;;
   ;; determine target's most detectable radiated frequecy
   ;;
   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
   (dolist (thisEnvRgn envRgnList)
     (when (ask environmentManager
           (objectInRegionp target (ask thisEnvRgn geoRegion)))
       (setq targetEnvRgn thisEnvRgn)
       (return)
       ))
   (setq targetEnvRgnName (ask targetEnvRgn envRgnName))
   (setq SF (first (getf (ask target
                     (platformAcousticParms-env-freq-list
                                 platformAcoustics))
                  targetEnvRgnName)))
   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

   (multiple-value-bind
     (PL AN sigma)
     (ask environmentManager (getPL-AN-Sigma searcher target))
     (let* ((RN  (if (< (platformStateParms-spd targetState)
                   (platformAcousticParms-spdThresh
                                 targetAcoustics))
              (platformAcousticParms-RNslow targetAcoustics)
              (platformAcousticParms-RNfast targetAcoustics)))
         (SN  (if (< (platformStateParms-spd searcherState)
                   (platformAcousticParms-spdThresh
                                 searcherAcoustics))
```

```
            (car  (getf (platformAcousticParms-SNlist
                                        searcherAcoustics) SF))
            (cadr (getf (platformAcousticParms-SNlist
                                        searcherAcoustics) SF))))
        (DI  (car   (getf (platformAcousticParms-DI-RD-list
                                        searcherAcoustics) SF)))
        (RD  (cadr  (getf (platformAcousticParms-DI-RD-list
                                        searcherAcoustics) SF)))
        (FOM (if (and RN SN AN DI RD)
             (- RN (+ (- (powerSum SN AN) DI) RD))))
        (MSE (if (and FOM PL)(- FOM PL)))

        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        ;;
        ;; using global and individual lambda-sigma-value
        ;; to adjust MSE
        ;;
        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        (SE  (if MSE
             (+ MSE (+ globalLambdaSigmaValue
                  (ask searcher
                  (platformAcousticParms-lambdaSigmaValue
                                        platformAcoustics))))))
        )
    (return-from getDetectionOppResult
              (if SE (if (< 0 SE) t nil) nil)
              )
    )
   )
  )
 )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (processTargetDetection Umpire) (ctime searcher searcherState
                                    target  targetState)
  (let ((reportDelay
      (ask searcher (getReportDelay
                  (platformStateParms-targetType targetState)))))
    (addEvent (make-coast-event
          :time     (+ cTime reportDelay)
          :object   (self)
          :procedure 'transmitDetectionReport
```

```
            :data      (list searcher target)
            :updateList (list searcher target)
          )
      )
      (when
         (and (or (equalp 'ASW-attack (ask searcher    mission))
                  (equalp 'ASW-trail  (ask searcher    mission)))
              (equalp 'submarineTarget (ask target   targetType))
         )
         (if (and hostilitiesp (equalp 'ASW-attack
                                       (ask searcher mission)))
          (processSubSubEngagement ctime reportDelay searcher
               searcherState target   targetState)
          (processTrackAndTrail ctime reportDelay searcher
               searcherState target targetState
                                       (ask searcher subTrailTime))
         )
      )
      (when
         (and (or (equalp 'ASuW-attack (ask searcher    mission))
                  (equalp 'ASuW-trail  (ask searcher    mission)))
              (equalp 'surfaceTarget   (ask target   targetType))
         )
         (if (and hostilitiesp (equalp 'ASuW-attack
                       (ask searcher mission)))
          (processSubSurfaceEngagement ctime reportDelay searcher
                       searcherState target targetState)
          (processTrackAndTrail  ctime reportDelay searcher
                       searcherState target targetState
                       (ask searcher surfaceTrailTime))
         )
      )
  )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (processTrackAndTrail Umpire) (cTime delay searcher  searcherState
target targetState trailTime)
 (when (and (ask searcher patrolRegion)(ask target patrolRegion))
  (let* ((theInterval  (+ (exponentialDraw trailTime) delay))
       )
      (ask searcher (beginTrail      ctime theInterval target))
```

```
      (ask target  (beginEngagement ctime theInterval searcher))
      (addEvent (make-coast-event
                    :time     (+ ctime theInterval)
                    :object   (self)
                    :procedure 'ProcessEndTrail
                    :data     searcher
                    :updateList (list searcher)
             )
      )
      (addEvent (make-coast-event
                    :time     (+ cTime theInterval)
                    :object   (self)
                    :procedure 'processPlatformDisengage
                    :data     target
                    :updateList (list target)
                 )
      )
      (recordMOEevent (make-MOEeventStruct
               :eventID :trailingPeriods
               :object1 (ask searcher MOEeventObjectIndex)
               :object2 (ask target  MOEeventObjectIndex)
               :data    (dround theInterval 3)))
  )
 )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (processSubSubEngagement Umpire) (ctime delay searcher
                                    searcherState target targetState)
 (let* ((targetClass  (ask target  (platformStateParms-classType
                                                  platformState)))
      (pkList (ask searcher (getf SubmarinePkAssocList
                                                  targetClass)))
      (searcherPK  (car pkList))
      (targetPK    (cadr pkList))
      (subSubEngagementTime (ask searcher
                                   subSubEngagementTime))
      (theInterval  (+ (exponentialDraw subSubEngagementTime)
                    delay))
     )
     (if (not (and (numberp searcherPK)(numberp targetPK)))
       (return-from processSubSubEngagement nil))
```

```
(if (ask searcher (or (< currentASWt ASWtPE)
                      (< currentCtMs CtMsPE)))
  (return-from processSubSubEngagement nil))

(setq searcherPK (ask target (expendCounterMeasures
                                                searcherPK)))
(setq targetPK   (ask target (expendASWtorpedos targetPK)))

(setq targetPK   (ask searcher (expendCounterMeasures
                                                targetPK)))
(setq searcherPK (ask searcher (expendASWtorpedos
                                                searcherPK)))

(let ((target-killed-p   (if (> searcherPK (random 1.0)) t nil))
      (searcher-killed-p (if (> targetPK   (random 1.0)) t nil))
     )
     (ask searcher (beginEngagement ctime theInterval target))
     (ask target   (beginEngagement ctime theInterval searcher))
     (if target-killed-p
       (addEvent (make-coast-event
                   :time      (+ ctime theInterval)
                   :object    (self)
                    :procedure 'processPlatformKill
                   :data      target
                   :updateList (list target)
                 )
       )
       (addEvent (make-coast-event
                   :time      (+ ctime theInterval)
                   :object    (self)
                    :procedure 'processPlatformDisengage
                   :data      target
                   :updateList (list target)
                 )
       )
     )
     (if searcher-killed-p
       (addEvent (make-coast-event
                   :time      (+ ctime theInterval)
                   :object    (self)
                    :procedure 'processPlatformKill
                   :data      searcher
                   :updateList (list searcher)
```

```
                )
            )
            (addEvent (make-coast-event
                        :time     (+ ctime theInterval)
                        :object   (self)
                        :procedure 'processPlatformDisengage
                        :data     searcher
                        :updateList (list searcher)
                      )
            )
          )
        )
    )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (processSubSurfaceEngagement Umpire) (ctime delay
                    searcher searcherState target targetState)
 (let ((ASuWweaponsRange (ask searcher ASuWweaponsRange)))
 (when
        (> ASuWweaponsRange (car (getRngBrg (platformStateParms-lat
searcherState)
                                (platformStateParms-lng searcherState)
                                (platformStateParms-lat targetState)
                                (platformStateParms-lng targetState)
                                *greatCircle*)))
    (ask searcher (resumePlan ctime interceptLegCount))
    (if (> (ask searcher currentASuWt) 0)
       (processSubSurfaceAttack ctime searcher searcherState
                                                target targetState))
    (return-from processSubSurfaceEngagement)
 )
 (multiple-value-bind
  (CPAdistance timeToCPA InterceptCourse)
  (calculateCPA searcherState targetState (ask searcher
                                                    interceptSpeed))
  (if (and  InterceptCourse
        (< timeToCPA  (ask searcher maxASuWInterceptTime))
       (< CPAdistance ASuWweaponsRange)
    )
    (let ((cpa (getlatlng (platformStateParms-lat searcherState)
                    (platformStateParms-lng searcherState)
```

```
                    (* timeToCPA (ask searcher interceptSpeed))
                    InterceptCourse
                    (platformStateParms-nav searcherState))))
(unless glb.batchp

        (drawTrack (list (list (platformStateParms-lat
                                              searcherState)
                       (platformStateParms-lng searcherState)
                    ) cpa))
        (sleep .5)
        (drawTrack (list (list (platformStateParms-lat
                                              searcherState)
                       (platformStateParms-lng searcherState)
                    ) cpa))
)
        (ask searcher (interruptPlan ctime Interceptcourse
                                    timeToCPA target))
    )
    (ask searcher (resumePlan ctime interceptLegCount))
 )
)
)
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (processSubSurfaceAttack Umpire) (ctime searcher searcherState
target targetState)
 (let ((theInterval (exponentialDraw (ask searcher
                              subSurfaceEngagementTime))))
  (ask searcher (processSurfaceAttack target targetState
                                              theInterval))
  (ask searcher (beginEngagement ctime theInterval target))
  (ask target   (beginEngagement ctime theInterval searcher))
  (addEvent (make-coast-event
          :time     (+ ctime theInterval)
          :object   (self)
          :procedure 'processPlatformDisengage
          :data     searcher
          :updateList (list searcher)))
  (addEvent (make-coast-event
          :time     (+ ctime theInterval)
          :object   (self)
```

```
            :procedure 'processPlatformDisengage
            :data      target
            :updateList (list target)))
 )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (transmitDetectionReport Umpire) (ctime platforms)
  (let* ((searcher (car  platforms))
         (target   (cadr platforms))
         (targetState (ask target platformState))
         (messageType (case (platformStateParms-targetType targetState)
                         (submarineTarget  'Sub-Detect-msg)
                         (surfaceTarget    'Surf-Detect-msg)))
        )
       (when (not messageType)
            (return-from transmitDetectionReport))
       (ask searcher
            (transmit  ctime
                   (make-coast-message
                       :send-time        ctime
                       :type           messageType
                       :content          (list (make-obu-report
                                                 ctime targetState))
                       :size            0
                       :transmission-path (list searcher)
                       :transmission-count 0)))
 )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (processEndTrail Umpire) (ctime platform)
  (ask platform (endTrail ctime))
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (processPlatformDisengage Umpire) (ctime platform)
  (ask platform (endEngagement ctime))
)
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; modified function
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (processPlatformKill Umpire) (ctime platform)
  (let* ((platformState (ask platform platformState))
         (platform-activep (platformStateParms-activep
                                                        platformState))
         )
    ; check if it was killed already
    (if (not platform-activep)
      (return-from processPlatformKill nil)
     )
    (ask platform (die))
    )
  )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (beginFalseAlarms Umpire) (cTime)
  (dolist (sideStruct (ask objectManager forceStructures))
    (dolist (searcher (getf (forceStruct-instanceList sideStruct)
                                              :FOMsearchers))
      (setNextFalseAlarm ctime searcher   'surfaceTarget)
      (setNextFalseAlarm ctime searcher 'submarineTarget))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; modified function
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (setNextFalseAlarm Umpire)(ctime searcher targetType)

  ; check if the searacher was killed
  (if (ask searcher (not (platformStateParms-activep platformState)))
       (return-from processFalseAlarm nil)
  )
  (if (>= 0 (ask searcher  computedFAR))
     (return-from setNextFalseAlarm))
   (addEvent
```

```
        (make-coast-event
            :time (+ ctime (ask searcher (exponentialDraw
                                          (/ 24 computedFAR))))
            :object  (self)
            :procedure 'processFalseAlarm
            :data  (list searcher targetType)
            :updateList (list searcher)
        )
    )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; modified function
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (processFalseAlarm Umpire)(ctime data)
 (let ((searcher   (car  data))
       (targetType (cadr data)))

  ; check if the searcher was killed
  (if (ask searcher (not (platformStateParms-activep platformState)))
       (return-from processFalseAlarm nil)
  )
  (if (ask searcher (platformStateParms-engagedp platformState))
     (progn
      (setNextFalseAlarm ctime searcher targetType)
      (return-from processFalseAlarm nil)))
  (let ((reportDelay (ask searcher (getReportDelay targetType))))

     (addEvent
        (make-coast-event
            :time      (+ cTime reportDelay)
            :object    (self)
            :procedure 'transmitFalseAlarmReport
            :data      data
            :updateList (list searcher)
            )
        )
     )
  (setNextFalseAlarm :time searcher targetType)
 )
```

```
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (transmitFalseAlarmReport Umpire) (ctime data)
 (let* ((searcher   (car  data))
       (targetType (cadr data))
       (uncertainty    #|(if (equalp 'submarineTarget targetType)
                        maxSubDetectionRange
                        maxSurfaceDetectionRange) SCL|# 30)
       (platformState (ask searcher PlatformState))
       (lat          (platformStateParms-lat platformState))
       (lng          (platformStateParms-lng platformState))
       (messageType (case targetType
                     (submarineTarget 'Sub-Detect-msg)
                     (surfaceTarget    'Surf-Detect-msg)))
    )
    (ask searcher
     (transmit  ctime
     (make-coast-message
       :send-time         ctime
       :type              messageType
       :content           (list (make-false-obu-report
                                lat lng uncertainty ctime targetType))
       :size              0
       :transmission-path (list searcher)
       :transmission-count 0)))
 )
)
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; CHANGE LOG:
;;
;;  new data fields and input dialogs added for individual
;;  lambda-sigma process:
;;    individualSigma
;;    lambdaSigmaValue
;;    meanInterval
;;
;;  glimpse interval deleted now using target's detectable interval
;;    to schedule next detection event:
;;    detectableInterval
;;
;;  editEnv-Freq added for inputting environment/frequency pairs
;;  of a acousticplatform
;;
;;  setForStart modified
;;    initializes env-freq-list, lambda-sigma process parameters,
;;    target's detectable interval and schedules event for updating
;;    individual lambda-sigma value
;;
;;  updateLambdaSigmaValue added
;;    update lambda-sigma value and schedule event for next update
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(setq acousticPlatform (kindof generalPlatform))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; modified function
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (exist acousticPlatform) (init-list)
  (usual-exist init-list)
  (have 'platformAcoustics (make-platformAcousticParms))
  (have 'spdThresh  12.5)
  (have 'RNslow          135)
  (have 'RNfast          150)
  (have 'sigFreq         nil)
  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
(have 'individualSigma   0)
(have 'lambdaSigmaValue        0)
(have 'meanInterval                  1000) ;(=1/lambda)
(have 'detectableInterval  1) ;platform's detectable interval
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(have 'SNlist                ())
(have 'DI-RD-list  ())
(push 'SNlist savedVarNames)
(push 'DI-RD-list savedVarNames)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(have 'env-freq-list ())
(push 'env-freq-list savedVarNames)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(have 'dataNameList (append dataNameList
              '(:LambdaSigmaProcess
                (IndividualSigma meanInterval detectableInterval)
               :acousticSig
                ( spdThresh RNslow RNfast))))
(have 'dataValueList (append dataValueList
              `(:LambdaSigmaProcess
                (,individualSigma ,meanInterval ,detectableInterval)
               :acousticSig
                ( ,spdThresh ,RNslow ,RNfast))))
(have 'dataTypeList (append dataTypeList
              '(:LambdaSigmaProcess (data data data)
               :acousticSig (data data data))))
(have 'dataTextList (append dataTextList
              '(:LambdaSigmaProcess
                    ("Individual Sigma" "Mean Interval(1/Lambda)"
                                                "Detectable Int.")
               :acousticSig ("Speed Thresh."
                    "Rad Noise-slow" "Rad Noise-fast"
                    ))))
(have 'dataTemplateText (append dataTemplateText
              '(:LambdaSigmaProcess
                 ((" Decibels" "[0 - 20]")
                  (" Hours" "[0.25 - unlimited]")
                  (" Hours" "[0.25 - unlimited]"))
                :acousticSig
                 (
                  ("Knots"          "[0 - 30]")
```

```
                    ("Decibles"        "[0 - 300]")
                    ("Decibles"        "[0 - 300]")))))
 (let* ((rb-designators  '(LambdaSigmaProcess-rb acousSig-rb
                    SN-rb DI-RD-rb env-freq-rb frequency-rb))
      (rb-values (list dummyRBdata dummyRBdata dummyRBdata
                       dummyRBdata dummyRBdata sigFreq))
      (rb-names '(dummyRBdata dummyRBdata dummyRBdata
                       dummyRBdata dummyRBdata sigFreq))
      )
      (update-rb-lists rb-designators rb-values rb-names)
 )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (editSN acousticPlatform) ()
 (setq SNlist
  (inputPairs (getsearchFreqList (forceStruct-opponentList
                                                  sideStruct)) SNlist
          (list "self noise by frequency" "freq" "slow spd" "fast spd")))
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (editDI-RD acousticPlatform) ()
 (setq DI-RD-list
  (inputPairs (getsearchFreqList (forceStruct-opponentList
       sideStruct)) DI-RD-list (list "DI and RD by frequency"
                                                    "freq" "DI" "RD")))
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; new function for inputting environment frequency pair
;; of a acoustic platform
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (editEnv-Freq acousticPlatform) ()
  (setq env-freq-list
      (inputPairs (ask environmentManager (getEnvRgnNames))
        env-freq-list (list "frequency in environments"
                            "environment" "frequency" "") t))
```

```
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; modified function
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (setForStart acousticPlatform) ()
 (usual-setForStart)
 (setf (platformAcousticParms-sigFreq  platformAcoustics)
                                                    sigFreq
     (platformAcousticParms-spdThresh  platformAcoustics)
                                                    spdThresh
     (platformAcousticParms-RNslow    platformAcoustics)
                                                    RNslow
     (platformAcousticParms-RNfast    platformAcoustics)
                                                    RNfast
     (platformAcousticParms-SNlist    platformAcoustics)
                                                    SNlist
     (platformAcousticParms-DI-RD-list platformAcoustics)
                                                    DI-RD-list

     (platformAcousticParms-env-freq-list platformAcoustics)
                                                    env-freq-list
     (platformAcousticParms-individualSigma platformAcoustics)
                                                    individualSigma
     (platformAcousticParms-lambdaSigmaValue
                                   platformAcoustics)
                                   (* individualSigma (normalDraw))
     (platformAcousticParms-meanInterval platformAcoustics)
                                              meanInterval
     (platformAcousticParms-detectableInterval
                                   platformAcoustics)
                                              detectableInterval
     )
 ; schedule next update event of lambda-sigma value
 (addEvent (make-coast-event
           :time (+ (getCurrenttime) (exponentialDraw (platformAcousticParms-
meanInterval platformAcoustics)))
         :object (self)
         :procedure 'updateLambdaSigmaValue
         :data nil
```

```
          :updateList nil)
         )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; new function
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (updateLambdaSigmaValue acousticPlatform) (ctime data)
 (setf (platformAcousticParms-lambdaSigmaValue
                                platformAcoustics)
    (* (platformAcousticParms-individualSigma
         platformAcoustics) (normalDraw)))
 (addEvent (make-coast-event
           :time (+ (getCurrenttime) (exponentialDraw (platformAcousticParms-
meanInterval platformAcoustics)))
          :object (self)
          :procedure 'updateLambdaSigmaValue
          :data nil
          :updateList nil)
         )
 )
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; CHANGE LOG:
;;
;; makeMPAassignments modified
;;   the selection of next cue is the cue with the largest ratio of
;;   (MPA's time on station/SPA size);
;;
;; getSearchValues modified
;;   searachValues now contain
;;     searchTime (MPA's time on station)
;;     missionCount
;;     MPA
;;     SPA size
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(setq ASWOC (kindof command))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (exist ASWOC) (init-list)
  (usual-exist init-list)
  (have 'allocationInterval  2)
  (have 'allocationStartTime        0)
  (have 'MPAareaSearchMax           0)
  (have 'minTstaInitial             4)
  (have 'minTstaDiversion  2)
  (have 'maxSPAsize                 50000)
  (have 'assumedSearcherNav         *rhumbline*)
  (have 'assumedTargetVel                 10)
  (have 'assumedTargetTau                 4)
  (have  'allocationInterval              12)
  (have 'operatingRegion)
  (have 'AreaSearchRegion)
  (have 'currentAssignmentList)
  (have 'dataNameList (append dataNameList
                  '(:ASWOCallocationParms
                     (allocationStartTime allocationInterval
    MPAareaSearchMax
              maxSPAsize minTstaInitial minTstaDiversion))))
  (have 'dataValueList (append dataValueList
                  `(:ASWOCallocationParms
```

```
                (,allocationStartTime ,allocationInterval
              ,MPAareaSearchMax, maxSPAsize ,minTstaInitial
              ,minTstaDiversion))))
 (have 'dataTypeList (append dataTypeList
                '(:ASWOCallocationParms
              (data data data data data data))))
 (have 'dataTextList (append dataTextList
                '(:ASWOCallocationParms
                      ("ALLOC START TIME" "ALLOC INTERVAL"
                      "#MPAs AREA SEARCH" "MAX SPA SIZE"
                            "MIN TSTA ASSIGNED" "MIN TSTA
DIVERTED"))))
 (have 'dataTemplateText (append dataTemplateText
                '(:ASWOCallocationParms
                  (("Hours"       "[0 - 72]")
                   ("Hours"       "[0 - 72]")
                   ("MPAs"        "[0 - 100]")
                   ("SQR NM"      "[0 - 100,000]")
                   ("Hours"       "[0 - 72]")
                   ("Hours"       "[0 - 72]")
                   ))))

 (let* ((rb-designators  '(Operating-Region-rb Area-Search-Region-
                           rb ASWOCallocation-rb))
      (rb-values (list operatingRegion AreaSearchRegion
                           dummyRBdata))
      (rb-names       '(operatingRegion AreaSearchRegion
                           dummyRBdata))
     )
     (update-rb-lists rb-designators rb-values rb-names)
 ))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (setForStart ASWOC) ()
 (usual-setForStart)
 (setq currentAssignmentList ())
 (if (not operatingRegion) (return-from setForStart))

 (addEvent (make-coast-event
         :time (+ (getCurrentTime) allocationStartTime
               (- 0.1 (random 0.2)) allocationInterval)
         :object  (self)
```

```
        :procedure 'processMPAAllocation
        :data  (+ (getCurrentTime) allocationStartTime
                                   allocationInterval)
        :updateList nil)
 )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (processMPAAllocation ASWOC) (ctime data)
 (addEvent (make-coast-event
        :time (+ data (- 0.1 (random 0.2)) allocationInterval)
        :object  (self)
        :procedure 'processMPAAllocation
        :data  (+ data allocationInterval)
        :updateList nil)
 )
 (resetCurrentAssignmentList ctime)
 (let ((searchValues     (getSearchValues
                    ctime
                    (getUnassignedCues)
                    (append (getAvailableMPAs)
                          (getDivertableMPAs))))
    )
    (makeMPAassignments ctime searchValues)
 )
 (resetAreaSearch ctime)
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (getUnassignedCues ASWOC) ()
 (let ((allCues (getTacticalPicture 'Subpicture operatingRegion))
     (unassignedCues ()))
    (dolist (cue allCues)
      (if (isOpenCuep cue) (push cue unassignedCues))
    )
    (return-from getUnassignedCues unassignedCues)
 )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
(defobfun (isOpenCuep ASWOC) (cue)
 (let ((open t))
  (dolist (Assignment currentAssignmentList)
   (when open
    (let ((assignedCue (cadr Assignment)))
     (when (< (car (getRngBrg (car  assignedCue)
                              (cadr assignedCue)
                              (car cue)
                              (cadr cue)
                              *greatCircle*))
              (+ (sqrt (/ (caddr        cue) pi))
                 (sqrt (/ (caddr assignedCue) pi))))
      (setq open nil)
     )
    )
   )
  )
  (return-from isOpenCuep open)
 )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (resetCurrentAssignmentList ASWOC) (ctime)
 (let ((newList ()))
    (dolist (assignment currentAssignmentList)
     (if (< ctime (car assignment))
        (push assignment newList)))
    (setq currentAssignmentList newList)
 )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (getAvailableMPAs ASWOC) ()
 (let ((thisAswoc (self))
    (availableMPAlist nil) )
    (mapc #'(lambda (x)
          (mapc #'(lambda (y) (push y availableMPAlist))
                (ask x (getAvailableMPAs thisAswoc))))
         (getf (forceStruct-instanceList sideStruct) :ownMPAs))
    (return-from getAvailableMPAs availableMPAlist)
 ))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (getDivertableMPAs ASWOC) ()
 (let ((thisAswoc (self))
      (divertableMPAlist nil) )
     (mapc #'(lambda (x)
            (mapc #'(lambda (y) (push y divertableMPAlist))
                  (ask x (getDivertableMPAs thisAswoc))))
           (getf (forceStruct-instanceList sideStruct) :ownMPAs))
     (return-from getDivertableMPAs divertableMPAlist)
 )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (getMPAsOnAreaSearch ASWOC) ()
 (let ((thisAswoc (self))
      (availableMPAlist nil) )
     (mapc #'(lambda (x)
            (mapc #'(lambda (y) (push y availableMPAlist))
                  (ask x (getMPAsOnAreaSearch thisAswoc))))
           (getf (forceStruct-instanceList sideStruct) :ownMPAs))
     (return-from getMPAsOnAreaSearch availableMPAlist)
 )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; modified function
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (getSearchValues ASWOC) (ctime tracks AvailableMPAs)
 (let* ((trackCount     (list-length tracks))
       (MPACount       (list-length   AvailableMPAs))
       (searchValues (make-array (list trackCount MPACount))))
  (dotimes
   (j MPACount)
   (let* ((MPA          (nth j AvailableMPAs))
         (missionCount   (MPAstruct-missionCount MPA))
         (currentLat     (MPAstruct-lat     MPA))
         (currentLng     (MPAstruct-lng     MPA))
```

```
    (flightSpeed   (ask (MPAstruct-squadron MPA) flightSpeed))
    (missionTime   (if (MPAstruct-searchingp MPA)
                   (- (MPAstruct-missionEndTime MPA) ctime)
                   (ask (MPAstruct-squadron MPA) missionTime)))
    (timeThresh    (if (MPAstruct-searchingp MPA)
                   minTstaDiversion
                   minTstaInitial))
   )
(dotimes
  (i trackCount)
  (let* ((track      (nth i tracks))
         (spaLat     (car  track))
         (spaLng     (cadr  track))
         (spaArea    (caddr  track))
         (SPArange   (car (getRngBrg  currentLat currentLng
                                spaLat    spaLng
                                assumedSearcherNav)))
         (transitTime (/ SPArange flightSpeed))
         (ReturnRange (car (getRngBrg  (MPAstruct-baseLat MPA)
                                (MPAstruct-baseLng MPA)
                                spaLat    spaLng
                                assumedSearcherNav)))
         (returnTime (/ ReturnRange flightSpeed))
         (searchTime (- missiontime (+ transitTime returnTime)))
        )
        (if (< timeThresh searchTime)
          (let ((effArea
                  (+ spaArea
                    (expandAOUpos
                     assumedTargetVel
                     assumedTargetTau
                      (+ transitTime  (/ searchTime 2)))))
               )
               (if (< effArea maxSPAsize)
                 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                 ;;
                 ;; searachValues now contain
                  ;;  searchTime (MPA's time on station)
                  ;;  missionCount
                  ;;  MPA
                  ;;  SPA size
                  ;;
                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
                    (setf (aref searchValues i j)
                        (list searchTime
                            missionCount
                            MPA
                            (list spaLat spaLng effArea)
                            ))
                ))))))
    )
    (return-from getSearchValues searchValues)
  )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; modified function
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (makeMPAassignments ASWOC) (ctime searchValues)
 (let* ((dimensions     (array-dimensions searchValues))
       (trackCount     (car  dimensions))
       (MPACount       (cadr dimensions))
       (assignmentList ())
      )
  (do ((doneCueing nil)
       (nextCue    nil nil)
       (MPA nil)(theCue nil)(iCue nil)(jCue nil))
      (doneCueing ())
    (setq doneCueing t)
    (dotimes
      (i trackCount)
      (dotimes
        (j MPACount)
        (if (aref searchValues i j)
          (if  nextCue
            ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
            ;;
            ;; the selection of cue is decided on the ratio of
            ;; MPA's time on station/SPA-size
            ;;
            ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
            (when (or (> (/ (car (aref searchValues i j)) (caddr (cadddr
                    (aref searchValues i j))))
```

```
                    (/ (car nextCue) (caddr (cadddr nextCue))))
                  (and (= (/ (car (aref searchValues i j))
                         (caddr  (cadddr  (aref  searchValues  i  j))))
                     (/ (car nextCue) (caddr (cadddr
                                                    nextCue))))
                   (<  (cadr (aref searchValues i j))
                       (cadr  nextCue)))
            )
            (setq MPA    (caddr  (aref searchValues i j)))
            (setq theCue  (cadddr (aref searchValues i j)))
            (setq nextCue (aref searchValues i j))
            (setq iCue i)(setq jCue j)
            )
         ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
         (progn
          (setq MPA    (caddr  (aref searchValues i j)))
          (setq theCue  (cadddr (aref searchValues i j)))
          (setq nextCue (aref searchValues i j))
          (setq iCue i)(setq jCue j))
       )
      )
     )
    )
  (when nextCue
     (setq doneCueing nil)
     (push (list MPA theCue) assignmentList)
     (dotimes (i trackCount) (setf (aref searchValues i jCue) nil))
     (dotimes (j   MPACount) (setf (aref searchValues iCue j) nil)))
 )
 (dolist (assignment assignmentList)
  (let ((MPA (car  assignment))
       (cue (cadr assignment)))
   (setq currentAssigmentList
        (remove-if #'(lambda(x)(equalp x (MPAstruct-keepOutOrder
                                                       MPA)))
                currentAssignmentList))
  (ask (MPAstruct-squadron MPA) (dispatchMPAtoCue ctime MPA
                                                       cue))
  (push (list (MPAstruct-missionEndTime MPA) cue)
                                     currentAssignmentList)
  ))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
(defobfun (resetAreaSearch ASWOC) (ctime)
 (let* ((AvailableMPAs    (getAvailableMPAs))
        (MPAsOnAreaSearch  (getMPAsOnAreaSearch))
        (searchRegion      areaSearchRegion)
       )
  (sort availableMPAs #'(lambda(x y)
                          (< (MPAstruct-missionCount x)
                            (MPAstruct-missionCount y))))
  (dotimes
   (i (min (- MPAareaSearchMax (list-length MPAsOnAreaSearch))
           (list-length AvailableMPAs)))
    (let* ((MPA        (nth i availableMPAs))
           (searchArea (ask (MPAstruct-squadron MPA)
                                    uncuedSearchArea))
           (cuePoint   nil)
           (cue        nil)
           (openCuep   nil)
          )
          (dotimes (i 20)
           (when (not openCuep)
            (setq cuePoint (getRandomPointInRegion searchRegion))
            (setq cue      (list (car cuePoint) (cadr cuePoint)
                                 searchArea))
            (setq openCuep (isOpenCuep cue))
           )
           (if (and (= i 19)(not openCuep))(print "overlap"))
          )
          (ask (MPAstruct-squadron MPA) (dispatchMPAtoAreaSearch
                                    ctime MPA cue searchRegion))
          (push (list (MPAstruct-missionEndTime MPA) cue)
                                    currentAssignmentList)
    )
  )))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; CHANGE LOG:
;;
;; new data field added in MPAstruct
;;   engaged
;;
;; dispatchMPAtoAreaSearch and dispatchMPAtoCue modified
;;   schedules detection opportunity for the MPA to every possible
;;   submarine target based on the inversion of detection rate to the ;;   target;
;; setNextFalseAlarm added to schedule FA event
;;
;; endMPAflight modified
;;   reset MPAstruct-engagedp flag to NIL after finish its flight
;;
;; beginSearch modified
;;   schedules detection opportunity for the MPAQ to every possible
;;   submarine target based on the inversion of detection rate to the ;;   target;
;;
;; setNextDetectionOpportunity added
;;   glimpse rate detetection model used; now target's radiated
;;   frequency is used to get DI, RD, NP, sweepWidth;
;;   then mean glimpse interval is computed
;;   by inversing the detection rate(NVW/A);
;;   excludes submarine target not inside search area before MPA
;;   search ends depending upon range: begin random glimpsing,
;;   or calculate earliest possible time these platforms can come
;;   within range (based on maxDectionRange MaxClosingSpeed and
;;   GlimpseInterval); engaged MPA won't have further detection on
;;   other targets
;;
;; processDetectionOpportunity added
;;   if MPA is already engaged then it schedule next detection
;;   opportunity based on target's acoustic parameters (i.e. radiated
;;   frequency, DI, RD, NP) to get sweepWidth and calculate detection
;;   rate of target, then invert the detection rate to get glimpse
;;   interval; otherwise it will check if the target is detected by it
;;
;; targetDetectedp and targetcoveredp added
;;   check if the target is inside the MPA search circle
;;
;; setNextFalseAlarm added
;;   schedule false alarm event during MPA on station time
```

```
;;  by making a exponential draw from mean time between false
;;  alarm (/ 24 computedFAR)
;;
;; processFalseAlarm added
;;  if the MPA is still on station and not being killed it will transmit
;;  a false target message, schedule next false alarm, expend buoys
;;  for target classification
;;
;; endEngagement added
;;  reset MPAstruct-engagedp flag to NIL
;;
;; transmitFalseAlarmReport added
;;  transmit the false alarm report to its command
;;
;; localizeTarget modified
;;  check if the MPA is on station and the target is active
;;  before doing anything
;;
;; processDetectionEvent modified
;;  check if MPA is on station and the target is active
;;  before doing anything
;;
;; prosecuteTarget added
;;  if time permit and still carry enough weapons MPA will
;;  stay for another run after it finish prosecution
;;
;; loseMPA modified
;;  check if the MPA has not being killed yet
;;
;; getSubsInSearchArea MPAsquadron deleted
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(setq MPAsquadron (kindof communicator detectionReporter))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (exist MPAsquadron) (init-list)
 (usual-exist init-list)
 (have 'commander)
 (have 'commandObject)
 (have 'commandIndex (symbol-value (getf init-list
                                        :commandIndex)))
```

```
(have 'MPAlist)
(have 'subList)
(have 'SubmarinePkAssocList ())
(push 'SubmarinePkAssocList savedVarNames)
(have 'squadronSize              9)
(have 'missionTime               11)
(have 'turnaroundTime   6)
(have 'failureTime               100)
(have 'repairTime                48)
(have 'prosecutionTime   2)
(have 'channelsMonitored         16)
(have 'flightSpeed       200)
(have 'uncuedSearchArea          10000)
(have 'assumedSearcherNav *rhumbline*)
(have 'buoyLoadOut                       50)
(have 'buoysPerSearch            19)
(have 'buoysPerEngagement        4)
(have 'torpedoLoadOut            4)
(have 'torpedosPerEngagement  2)
(have 'DI-RD-list                ())
(push 'DI-RD-list                savedVarNames)
(have 'dataNameList (append dataNameList
                '(:MPAsquadron
                  (squadronSize missionTime turnaroundTime
              failureTime repairTime prosecutionTime
              channelsMonitored flightSpeed uncuedSearchArea )
                  :MPAStores
                   (buoyLoadOut buoysPerSearch buoysPerEngagement
                    torpedoLoadOut torpedosPerEngagement))))
(have 'dataValueList (append dataValueList
                `(:MPAsquadron
                   (,squadronSize ,missionTime ,turnaroundTime
              ,failureTime ,repairTime ,prosecutionTime
              ,channelsMonitored ,flightSpeed
              ,uncuedSearchArea)
                   :MPAStores
                    (,buoyLoadOut ,buoysPerSearch
              ,buoysPerEngagement, torpedoLoadOut
              ,torpedosPerEngagement))))
(have 'dataTypeList (append dataTypeList
                  '(:MPAsquadron
                   (data data data data data data data data data)
                   :MPAStores
```

```
                (data data data data data))))
(have 'dataTextList (append dataTextList
                '(:MPAsquadron
                  ("Squadron Size" "Mission Time" "Turnaround Time"
                   "Exp Time to Fail" "Exp Time to Repair" "Exp
              Prosecution Time" "Total Channels" "Transit Speed"
              "Uncued Search Area")
                  :MPAStores
                  ("Buoy Loadout" "Buoys / Search" "Buoys /
              Localization" "Torpedo Loadout"
              "Torpedo / Engagement"))))
(have 'dataTemplateText (append dataTemplateText
                '(:MPAsquadron
                  (("Integer"      "[0 - 50]")("Hours"       "[0 - 500]")
                   ("Hours"       "[0 - 500]")("Hours"       "[0 - 500]")
                   ("Hours"       "[0 - 500]")("Hours"       "[0 - 24]")
                   ("Integer"     "[0 - 50]")("NM per Hr"    "[0 - 24]")
                   ("NM squared"    "[0 - 50000]"))
                  :MPAStores
                  (("Integer"        "[0 - 100]")
                   ("Integer"        "[0 - 100]")
                   ("Integer"        "[0 - 100]")
                   ("Integer"        "[0 - 100]")
                   ("Integer"        "[0 - 100]")))))
(setf  (getf dataTextList :reports)
     '("REPORT TYPE" "POSITION ERROR" "SPEED ERROR"
             "COURSE ERROR" "Exp Localization Delay"
             "False Alarm Rate" "Track Initiation"))
(setf (getf dataTemplateText :reports)
    '(("POS or POS+VEL" "[POS , POS+VEL]")
     ("Nautical Miles" "[0 - 1000]")
     ("Knots"         "[0 - 30]")
     ("Degrees"       "[0 - 180]")
     ("Hours"         "[0 - 100]")
     ("FA per Hour"    "[0 - 1 ??]")
     ("Single Contact to Track"    "0 - no, 1 - yes")))

(let* ((rb-designators  '(MPAsquadron-rb MPAStores-rb
            commander-rb DI-RD-rb MPASubPk-rb ))
     (rb-values (list dummyRBdata dummyRBdata commander
            dummyRBdata dummyRBdata ))
     (rb-names '(dummyRBdata dummyRBdata commander
            dummyRBdata dummyRBdata ))
```

```
      )
      (update-rb-lists rb-designators rb-values rb-names)
 )
 (remove-rbs '(location-rb))
 (if (getf init-list :isClass)(remove-rbs '(commander-rb
                                            fusionCenter-rb)))
  (when (not (getf init-list :isClass))
    (remf C3-connectivities 'Surf-Detect-msg)
    (remf C3-connectivities 'Sub-SI-msg)
    (remf C3-connectivities 'Surf-SI-msg)
    (update-rb-lists '(Connect-rb) (list C3-connectivities)
                                            '(C3-connectivities)))
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (editDI-RD MPAsquadron) ()
 (setq DI-RD-list
  (inputPairs (getsearchFreqList (forceStruct-opponentList
    sideStruct)) DI-RD-list (list "DI and RD by frequnecy"
                                            "freq" "DI" "RD")))
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (editMPASubPKs MPAsquadron) ()
 (let ((classList
       (getClassesFromIndices
        (getf (forceStruct-objectKeyList sideStruct)
                                    :submarineTargets))))
   (setq SubmarinePkAssocList
       (inputPairs classList SubmarinePkAssocList
                (list "pk vs. sub class" "class" "pk" "target pk")))
 )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (setForStart MPAsquadron) ()
 (usual-setForStart)
 (setq commandObject (getObjectFromName
                    commander
                    commandIndex
```

```
                    (platformStateParms-side platformState)))
(have 'subTargetList (getf (forceStruct-instanceList sideStruct)
                           :submarineTargets))
(if (not commandObject)(return-from setForStart nil))
(setq MPAlist nil)
(dotimes (i squadronSize)
  (push (make-MPAstruct
        :ASWOC       commandObject
        :squadron    (self)
        :baseLat     (ask commandObject
                     (platformStateParms-lat platformstate))
        :baseLng     (ask commandObject
                     (platformStateParms-lng platformstate))
        :lat         (ask commandObject
                     (platformStateParms-lat platformstate))
        :lng         (ask commandObject
                     (platformStateParms-lng platformstate))
        :buoyCount   buoyLoadOut
        :torpedoCount torpedoLoadOut
        :failureTime (exponentialDraw  failureTime)
       )
       MPAlist
  )
 )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; modified function
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (dispatchMPAtoAreaSearch MPAsquadron) (ctime MPA cue
areaSearchRegion)
 (let* ((targetLat       (car  cue))
        (targetLng       (cadr  cue))
        (targetRadius    (sqrt (/ (caddr cue) pi)))
        (targetRange     (car (getRngBrg  (MPAstruct-lat MPA)
                         (MPAstruct-lng MPA) targetLat targetLng
                                  assumedSearcherNav)))
        (transitTime     (/ targetRange flightSpeed))
        (ReturnRange     (car (getRngBrg  (MPAstruct-baseLat MPA)
                                  (MPAstruct-baseLng MPA)
```

```
                                targetLat    targetLng
                                assumedSearcherNav)))
        (returnTime      (/ ReturnRange flightSpeed))
        (searchStartTime  (+ ctime transitTime))
        (missionEndTime   (+ ctime missionTime))
        (searchEndTime    (- missionEndTime returnTime))
        (flightHours      (+ (MPAstruct-totalFlightHours MPA)
                          missionTime))
        (misssionCount    (+ (MPAstruct-missionCount MPA) 1))
       )
(unless glb.batchp
    (if (MPAstruct-drawnMission MPA)
       (plotCueingPicture
         (nth 0 (MPAstruct-drawnMission MPA))
         (nth 1 (MPAstruct-drawnMission MPA))
         (nth 2 (MPAstruct-drawnMission MPA))
         (nth 3 (MPAstruct-drawnMission MPA))
         (nth 4 (MPAstruct-drawnMission MPA))))
    (plotCueingPicture
     (MPAstruct-lat MPA)
     (MPAstruct-lng MPA)
     targetLat
     targetLng
     targetRadius))
    (setf (MPAstruct-drawnMission    MPA) (list
                                (MPAstruct-lat MPA)
                                (MPAstruct-lng MPA)
                                targetLat
                                targetLng
                                targetRadius)
    )
    (setf (MPAstruct-searchingp      MPA) t
        (MPAstruct-searchStartTime  MPA) searchStartTime
        (MPAstruct-searchEndTime    MPA) searchEndTime
        (MPAstruct-missionEndTime   MPA) missionEndTime
        (MPAstruct-lat              MPA) targetLat
        (MPAstruct-lng              MPA) targetLng
        (MPAstruct-searchRegion     MPA) areaSearchRegion
        (MPAstruct-radius           MPA) targetRadius
        (MPAstruct-totalFlightHours MPA) flightHours
       (MPAstruct-missionCount     MPA) misssionCount
    )
    (setf (MPAstruct-keepOutOrder MPA) (list
```

```
                        (MPAstruct-missionEndTime MPA) cue))
    (addEvent (make-coast-event
            :time (+ ctime transitTime)
            :object (self)
            :procedure 'beginSearch
            :data MPA
            :updateList subTargetList))
    (addEvent (make-coast-event
            :time (+ ctime transitTime)
            :object (self)
            :procedure 'setNextFalseAlarm
            :data MPA
            :updateList nil))
    (addEvent (make-coast-event
            :time missionEndTime
            :object (self)
            :procedure 'endMPAflight
            :data MPA
            :updateList nil))
)
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; modified function
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (dispatchMPAtoCue MPAsquadron) (ctime MPA target)
 (let* ((targetLat      (car  target))
        (targetLng      (cadr  target))
        (targetRadius    (sqrt (/ (caddr target) pi)))

        (targetRange     (car (getRngBrg  (MPAstruct-lat MPA)
                              (MPAstruct-lng MPA)
                              targetLat targetLng
                              assumedSearcherNav)))
        (transitTime     (/ targetRange flightSpeed))
        (ReturnRange      (car (getRngBrg  (MPAstruct-baseLat MPA)
                                  (MPAstruct-baseLng MPA)
                                  targetLat    targetLng
                                  assumedSearcherNav)))
        (returnTime      (/ ReturnRange flightSpeed))
```

```
        (searchStartTime  (+ ctime transitTime))
        (missionTime      (if (MPAstruct-searchingp MPA)
                          (- (MPAstruct-missionEndTime MPA) ctime)
                          missionTime))
        (missionEndTime   (+ ctime missionTime))
        (searchEndTime    (- missionEndTime returnTime))
        (flightHours      (if (MPAstruct-searchingp MPA)
                          (MPAstruct-totalFlightHours MPA)
                          (+ (MPAstruct-totalFlightHours MPA)
                          missionTime)))
        (misssionCount    (if (MPAstruct-searchingp MPA)
                          (MPAstruct-missionCount MPA)
                          (+ (MPAstruct-missionCount MPA) 1)))
    )
(unless glb.batchp
    (if (MPAstruct-drawnMission MPA)
      (plotCueingPicture
       (nth 0 (MPAstruct-drawnMission MPA))
       (nth 1 (MPAstruct-drawnMission MPA))
       (nth 2 (MPAstruct-drawnMission MPA))
       (nth 3 (MPAstruct-drawnMission MPA))
       (nth 4 (MPAstruct-drawnMission MPA))))
    (plotCueingPicture
     (MPAstruct-lat MPA)
     (MPAstruct-lng MPA)
     targetLat
     targetLng
     targetRadius))
    (setf (MPAstruct-drawnMission    MPA) (list
                                (MPAstruct-lat MPA)
                                (MPAstruct-lng MPA)
                                targetLat
                                targetLng
                                targetRadius)
    )
    (setf (MPAstruct-searchingp      MPA) t
        (MPAstruct-searchStartTime  MPA) searchStartTime
        (MPAstruct-searchEndTime    MPA) searchEndTime
        (MPAstruct-missionEndTime   MPA) missionEndTime
        (MPAstruct-lat              MPA) targetLat
        (MPAstruct-lng              MPA) targetLng
        (MPAstruct-radius           MPA) targetRadius
        (MPAstruct-searchRegion     MPA) nil
```

```
            (MPAstruct-totalFlightHours MPA) flightHours

           (MPAstruct-missionCount     MPA) misssionCount
      )
       (setf (MPAstruct-keepOutOrder MPA) (list (MPAstruct-missionEndTime
MPA) target))
     (addEvent (make-coast-event
                :time (+ ctime transitTime)
                :object  (self)
                 :procedure 'beginSearch
                :data  MPA
                :updateList subTargetList))
     (addEvent (make-coast-event
                :time (+ ctime transitTime)
                :object  (self)
                 :procedure 'setNextFalseAlarm
                :data MPA
                :updateList nil))
     (addEvent (make-coast-event
                :time missionEndTime
                :object  (self)
                 :procedure 'endMPAflight
                :data  MPA
                :updateList nil))
 )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (endMPAflight MPAsquadron) (ctime MPA)
  (if (or (not (MPAstruct-searchingp MPA))
        (MPAstruct-killedp MPA))
     (return-from endMPAflight nil))
 (unless glb.batchp
  (if (MPAstruct-drawnMission MPA)
         (plotCueingPicture
           (nth 0 (MPAstruct-drawnMission MPA))
           (nth 1 (MPAstruct-drawnMission MPA))
           (nth 2 (MPAstruct-drawnMission MPA))
           (nth 3 (MPAstruct-drawnMission MPA))
           (nth 4 (MPAstruct-drawnMission MPA)))))
  (setf (MPAstruct-lat MPA) (MPAstruct-baselat MPA)
      (MPAstruct-lng                     MPA)(MPAstruct-baseLng MPA)
```

```
      (MPAstruct-readyp                MPA) nil
      (MPAstruct-searchingp            MPA) nil
      (MPAstruct-searchStartTime       MPA) nil
      (MPAstruct-searchEndTime               MPA) nil
      (MPAstruct-missionEndTime              MPA) nil
      (MPAstruct-radius                MPA) nil
      (MPAstruct-searchRegion          MPA) nil
      (MPAstruct-drawnMission          MPA) nil
      (MPAstruct-keepOutOrder          MPA) nil
      (MPAstruct-nextContact           MPA) nil
      (MPAstruct-engagedp                    MPA) nil
 )
 (if (> (MPAstruct-totalFlightHours MPA)
        (MPAstruct-failureTime      MPA))
  (addEvent (make-coast-event
             :time (+ (getCurrentTime) turnaroundTime
                      (exponentialDraw  repairTime))
            :object  (self)
             :procedure 'repairMPA
            :data  MPA
            :updateList nil))
  (addEvent (make-coast-event
             :time (+ (getCurrentTime) turnaroundTime)
            :object  (self)
             :procedure 'readyMPA
            :data  MPA
            :updateList nil))
 )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (repairMPA MPAsquadron) (ctime MPA)
 (setf (MPAstruct-failureTime MPA)
       (+ (MPAstruct-totalFlightHours MPA)
          (exponentialDraw  failureTime))
 )
 (readyMPA ctime MPA)
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (readyMPA MPAsquadron) (ctime MPA)
```

```
 (setf (MPAstruct-readyp       MPA) t
      (MPAstruct-buoyCount    MPA) buoyLoadOut
      (MPAstruct-torpedoCount MPA) torpedoLoadOut
  )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (getAvailableMPAs MPAsquadron) (thisAswoc)
 (let ((availableMPAs ())
       )
       (dolist (x MPAlist)
             (if (and (not (MPAstruct-killedp x))
                   (equalp thisAswoc (MPAstruct-ASWOC x))
                    (MPAstruct-readyp x)
                    (not (MPAstruct-searchingp x)))
                (push x availableMPAs)))
       (return-from getAvailableMPAs availableMPAs)
 )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (getDivertableMPAs MPAsquadron) (thisAswoc)
 (let ((divertableMPAs ())
    )
  (dolist (x MPAlist)
    (if (and (not (MPAstruct-killedp x))
          (equalp thisAswoc (MPAstruct-ASWOC x))
          (MPAstruct-searchRegion x)
                  (>=  (MPAstruct-buoyCount  x)  (+  buoysPerSearch
buoysPerEngagement)))
      (push x divertableMPAs)))
  (return-from getDivertableMPAs divertableMPAs)
 )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (getMPAsOnAreaSearch MPAsquadron) (thisAswoc)
 (let ((availableMPAs ())
     )
  (dolist (x MPAlist)
```

```
    (if (and (not (MPAstruct-killedp x))
            (equalp thisAswoc (MPAstruct-ASWOC x))
            (MPAstruct-searchRegion x))
     (push x availableMPAs)))
   (return-from getMPAsOnAreaSearch availableMPAs)
  )
 )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (plotSelf MPAsquadron) ()
 (unless glb.batchp
 (if (and (platFormStateParms-activep platFormState)
        (clockIsOnp))
    (mapc #'(lambda(MPA)
             (if (MPAstruct-drawnMission MPA)
               (plotCueingPicture
                (nth 0 (MPAstruct-drawnMission MPA))
                (nth 1 (MPAstruct-drawnMission MPA))
                (nth 2 (MPAstruct-drawnMission MPA))
                (nth 3 (MPAstruct-drawnMission MPA))
                (nth 4 (MPAstruct-drawnMission MPA)))))
           MPAlist
    )
  )
 )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; modified function
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (beginSearch MPAsquadron) (ctime MPA)
  (expendBuoys   MPA   buoysPerSearch)
  (dolist (target subTargetList)
   (setNextDetectionOpportunity ctime (list MPA target))
   )
  )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
```

```
;; new function
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (setNextDetectionOpportunity MPAsquadron)(ctime data)
  (let* ((MPA (car data))
        (target (cadr data))
        (targetState (ask target platformState))
        (targetAcoustics (ask target platformAcoustics))
        )

   (if (or (not (MPAstruct-searchingp MPA))
        (MPAstruct-killedp MPA)
        (not (platformStateParms-activep targetState))
        )
    (return-from setNextDetectionOpportunity nil))

   #|
     glimpse rate detetection model used; now target's radiated
     frequency is used to get DI, RD, NP, sweepWidth
     then mean glimpse interval is computed by inversing the
      detection rate (NVW/A)
   |#

   (let* ((rng (car (getRngBrg (MPAstruct-lat MPA)
                        (MPAstruct-lng MPA)
                        (platformStateParms-lat targetState)
                        (platformStateParms-lng targetState)
                        assumedSearcherNav)))
        (separation (- rng (MPAstruct-radius MPA)))
        (maxClosingSpeed (ask umpire maxClosingSpeed));
        (RN  (if (< (platformStateParms-spd targetState)
                 (platformAcousticParms-spdThresh targetAcoustics))
              (platformAcousticParms-RNslow targetAcoustics)
              (platformAcousticParms-RNfast targetAcoustics)))
        (targetEnvRgn nil)
        (tagetEnvRgnName nil)
        (envRgnList (ask environmentManager envRegionList))

        (SF  nil)
        (DI  nil)
        (RD  nil)
        (NP  nil)
```

```
        (sweepWidth 0)
        glimpseInterval
        meanGlimpseInterval
        (searchTime 2)
        )

    (dolist (thisEnvRgn EnvRgnList)
     (when (objectInRegionp target (ask thisEnvRgn geoRegion))
        (setq targetEnvRgn thisEnvRgn)
        (return)
        ))
    (when targetEnvRgn
     (setq targetEnvRgnName (ask targetEnvRgn envRgnName))
        (setq SF (first (getf (ask target (platformAcousticParms-env-freq-list
platformAcoustics)) targetEnvRgnName)))
      (setq DI  (first   (getf DI-RD-list SF)))
      (setq RD  (second  (getf DI-RD-list SF)))
      (setq NP  (if (and DI RD) (- (+ RN DI) RD)))
      )
     (setq sweepWidth (if NP (ask environmentManager (getSweepWidth target
NP)) 0))
    (if (or (eql 0 sweepWidth)
          (eql 0 (*  (ask target (cond (track meanTransitSpeed)
                                       (patrolRegion meanPatrolSpeed)
                                       (t interceptSpeed)))
                     (min channelsMonitored buoysPerSearch)
                     sweepWidth))
          (eql 0 (* pi (MPAstruct-radius MPA)(MPAstruct-radius MPA))))
      (return-from setNextDetectionOpportunity nil)
      )
        (setq meanGlimpseInterval (/ 1 (/ (*   (ask target (cond (track
meanTransitSpeed)
                                                  (patrolRegion meanPatrolSpeed)
                                                  (t interceptSpeed)))
                                  (min channelsMonitored buoysPerSearch)
                                  sweepWidth)
                                   (* pi (MPAstruct-radius MPA)(MPAstruct-radius
MPA)))))
    (setq GlimpseInterval (exponentialDraw meanGlimpseInterval))

       (setq searchTime (- (MPAstruct-searchEndTime MPA) (MPAstruct-
searchStartTime MPA)))
```

```
#|
  exclude submarine target not inside search area before
  MPA search ends
|#
(when (and (< separation (* searchTime maxClosingSpeed))
        (< (+ cTime GlimpseInterval) (MPAstruct-searchEndTime
                                                    MPA))
        )

 #|
   check if MPA is engaged
 |#

 (when (MPAstruct-engagedp       MPA)
    (if (< 0 (- separation (* maxClosingSpeed GlimpseInterval)))
     (addEvent (make-coast-event
                :time (+ cTime (max GlimpseInterval
                                    (/ separation maxClosingSpeed)))
                :object  (self)
                :procedure 'setNextDetectionOpportunity
                :data  data
                :updateList (list (self) target))
       )
     (addEvent (make-coast-event
                :time (+ cTime GlimpseInterval)
                :object  (self)
                :procedure 'setNextDetectionOpportunity
                :data  data
                :updateList (list (self) target))
      )
     )
    (return-from setNextDetectionOpportunity nil)
    )
 #|
 depending upon range: begin random glimpsing,
 or calculate earliest possible time these platforms can come
 within range (based on maxDectionRange MaxClosingSpeed and
 GlimpseInterval)
 |#
 (if (< 0 (- separation (* maxClosingSpeed GlimpseInterval)))
    (addEvent (make-coast-event
            :time (+ cTime (max GlimpseInterval
                                (/ separation maxClosingSpeed)))
```

```
                    :object  (self)
                    :procedure 'setNextDetectionOpportunity
                    :data  data
                    :updateList (list (self) target))
                   )
            (addEvent (make-coast-event
                    :time (+ cTime GlimpseInterval)
                    :object  (self)
                    :procedure 'processDetectionOpportunity
                    :data  data
                    :updateList (list (self) target)
                    )
                   )
            )
           )
     )
     )
   )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; new function
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (processDetectionOpportunity MPAsquadron) (ctime data)
   (let* ((MPA (first data))
          (target (second data))
          (targetState     (ask target       platformState))
          (targetAcoustics (ask target   platformAcoustics)))
     (when (and (platformStateParms-activep   targetState)
                (MPAstruct-searchingp MPA)
                )
       (when (MPAstruct-engagedp MPA)
         (let* ((RN  (if (< (platformStateParms-spd targetState)
                            (platformAcousticParms-spdThresh targetAcoustics))
                         (platformAcousticParms-RNslow targetAcoustics)
                         (platformAcousticParms-RNfast targetAcoustics)))
                (targetEnvRgn nil)
                (tagetEnvRgnName nil)
                (envRgnList (ask environmentManager envRegionList))
                (SF  nil)
                (DI  nil)
```

```
    (RD  nil)
    (NP  nil)
    (sweepWidth 0)
    glimpseInterval
    meanGlimpseInterval )
  (dolist (thisEnvRgn EnvRgnList)
   (when (objectInRegionp target (ask thisEnvRgn geoRegion))
    (setq targetEnvRgn thisEnvRgn)
    (return)
    ))
  (when targetEnvRgn
   (setq targetEnvRgnName (ask targetEnvRgn envRgnName))
   (setq SF (first (getf (ask target
                    (platformAcousticParms-env-freq-list
                         platformAcoustics))
                    targetEnvRgnName)))
   (setq DI  (first   (getf DI-RD-list SF)))
   (setq RD  (second  (getf DI-RD-list SF)))
   (setq NP  (if (and DI RD) (- (+ RN DI) RD))))
  (setq sweepWidth (if NP (ask environmentManager
                         (getSweepWidth target NP)) 0))
  (if (or (eql 0 sweepWidth)
        (eql 0 (*  (ask target (cond (track meanTransitSpeed)
                         (patrolRegion meanPatrolSpeed)
                         (t interceptSpeed)))
                 (min channelsMonitored buoysPerSearch)
                 sweepWidth))
        (eql 0 (* pi (MPAstruct-radius MPA)(MPAstruct-radius
                                              MPA))))
    (return-from processDetectionOpportunity nil))
  (setq meanGlimpseInterval
 (/ 1 (/ (*  (ask target (cond (track meanTransitSpeed)
                         (patrolRegion meanPatrolSpeed)
                         (t interceptSpeed)))
              (min channelsMonitored buoysPerSearch)
              sweepWidth)
        (* pi (MPAstruct-radius MPA)(MPAstruct-radius
                                              MPA)))))
  (setq GlimpseInterval (exponentialDraw
                                        meanGlimpseInterval))

  (addEvent (make-coast-event
        :time (+ cTime (exponentialDraw glimpseInterval))
```

```
                :object (self)
                :procedure 'processDetectionOpportunity
                :data data
                :updateList (if target (list target))
                 ))
          )
          (return-from processDetectionOpportunity nil)
         )
        (if (targetDetectedp MPA target targetState targetAcoustics)
              (processDetectionevent ctime data))
        (setNextDetectionOpportunity ctime data)
         )
      )
    )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; new function
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (targetDetectedp MPAsquadron) (MPA target targetState
targetAcoustics)
 (if (not (targetCoveredp MPA target targetState))
     (return-from targetDetectedp nil)
     (return-from targetDetectedp t)
     )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; new function
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (targetCoveredp MPAsquadron)(MPA target targetState)
 (return-from targetCoveredp
  (< (car (getRngBrg (MPAstruct-lat MPA)
                     (MPAstruct-lng MPA)
                     (platformStateParms-lat targetState)
                     (platformStateParms-lng targetState)
                     assumedSearcherNav))
    (MPAstruct-radius MPA)
```

```
  )
 )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; new function
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (setNextFalseAlarm MPAsquadron) (cTime MPA)
   (if (>= 0  computedFAR)
     (return-from setNextFalseAlarm))
   (addEvent
    (make-coast-event
     :time (+ ctime (exponentialDraw (/ 24 computedFAR)))
     .object  (self)
     :procedure 'processFalseAlarm
     :data  MPA
     :updateList (list (self))
     )
    )
  )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; new function
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (processFalseAlarm MPAsquadron)(ctime MPA)
  (if (or (not (MPAstruct-searchingp MPA))
        (MPAstruct-killedp MPA))
      (return-from processFalseAlarm nil)
  )
  (if (MPAstruct-engagedp MPA)
     (progn
      (setNextFalseAlarm ctime MPA)
      (return-from processFalseAlarm nil)))
  (let ((reportDelay (getReportDelay 'submarineTarget)))
     (expendBuoys    MPA    buoysPerEngagement) ; expend buoys for
                                                ;target classification
     (addEvent
```

```
      (make-coast-event
            :time     (+ cTime reportDelay)
            :object   (self)
             :procedure 'transmitFalseAlarmReport
            :data     MPA
            :updateList (list (self))
            )
       )
      )
  (setNextFalseAlarm ctime MPA)
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; new function
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (endEngagement MPAsquadron) (ctime MPA)
   (setf (MPAstruct-engagedp MPA) nil))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; new functi on
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (transmitFalseAlarmReport MPAsquadron) (ctime MPA)
 ; check if the MPA still on station and not being killed
 (if (or (not (` fPAstruct-searchingp MPA))
       (MPAstruct-killedp MPA))
      (retu·n· from transmitFalseAlarmReport nil)
 )
 (let ((theReport (list (make-false-obu-report
          (MPAstruct-lat MPA)
          (MPAstruct-lng MPA)
          (MPAstruct-radius MPA)
          ctime 'submarineTarget)))
    )
  (transmit  ctime
        (make-coast-message
         :send-time        ctime
         :type            'Sub-Detect-msg
         :content          theReport
```

```
        :size          0
        :transmission-path (list (self))
        :transmission-count 0))
  )
 )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; modified function
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (ProcessDetectionEvent MPAsquadron) (ctime data)
 (let* ((MPA       (car  data))
     (target    (cadr data))
     (targetState (ask target platformState))
     )
  ; check if MPA is on station and the target is active
  (if (not (and (platformStateParms-activep targetState)
            (MPAstruct-searchingp MPA))
        )
    (return-from ProcessDetectionEvent nil))
  (setf (MPAstruct-searchRegion MPA) nil)
  (addEvent
   (make-coast-event
    :time      (min (+ ctime (getReportDelay))
               (MPAstruct-searchEndTime MPA))
    :object    (self)
    :procedure 'localizeTarget
    :data      data
    :updateList (if target (list target))
    ))
  )
 )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; modified function
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (localizeTarget MPAsquadron) (ctime data)
 (let* ((lself (self))
```

```
    (MPA      (car  data))
    (target   (cadr  data))
    (targetState  (ask target platformState))
    (theReport  (if target
                    (list (make-obu-report  ctime
                     (ask target platformState)))
                    (list (make-false-obu-report
                         (MPAstruct-lat    MPA)
                         (MPAstruct-lng    MPA)
                         (MPAstruct-radius MPA)
                         ctime 'Sub-Detect-msg)))))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; check if the MPA is on station and searching
;; the target is active
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(if (not (and (platformStateParms-activep targetState)
          (MPAstruct-searchingp MPA))
     )
   (return-from localizeTarget nil))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(transmit  ctime
      (make-coast-message
       :send-time        ctime
       :type             'Sub-Detect-msg
       :content          theReport
       :size             0
       :transmission-path (list (self))
       :transmission-count 0))
(when (or (< (MPAstruct-torpedoCount MPA) torpedosPerEngagement)
      (< (MPAstruct-buoyCount    MPA)    buoysPerEngagement))
   (endMPAflight ctime MPA)
    (return-from localizeTarget nil))

(when (and (hostilitiesp) target)
 (expendBuoys   MPA   buoysPerEngagement)
 (let ((engagementTime (exponentialDraw prosecutionTime)))
   (when (MPAstruct-searchEndTime MPA)
     (when (<  (+ cTime engagementTime) (MPAstruct-searchEndTime MPA))
       (ask target (beginEngagement ctime engagementTime lself))
       (setf (MPAstruct-engagedp MPA) t)
```

```
        (addEvent
         (make-coast-event
          :time      (+ cTime engagementTime)
          :object    (self)
           :procedure 'prosecuteTarget
          :data      data
          :updateList (if target (list target))
          ))
        )
       )
      )
     )
    )
  )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; modified function
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (prosecuteTarget MPAsquadron) (ctime data)
  (let* ((MPA         (car  data))
         (MPASide     (platformStateParms-side      platformState))
         (MPAType     (platformStateParms-classType platformState))
         (target  (cadr data))
         (targetState (ask target platformState))
         (targetSide  (platformStateParms-side targetState))
         (targetType (platformStateParms-classType targetState))
         (pklist      (getf SubmarinePkAssocList targetType))
         (searcherPK (ask target (expendCounterMeasures (car  pklist,
         (targetPK    (ask target (expendSubSams (cadr pklist))))
         (objectNumber  MOEeventObjectIndex)
        )

    (if (not (and (platformStateParms-activep targetState)
                  (MPAstruct-searchingp MPA))
             )
      (return-from prosecuteTarget nil))
    (when (not (and (numberp searcherPK)(numberp targetPK)))
          (ask target (endEngagement ctime))
           (return-from prosecuteTarget nil))
    (expendTorpedos MPA torpedosPerEngagement)
```

```
(when (> targetPK   (random 1.0))
 (loseMPA ctime MPA (ask target MOEeventObjectIndex)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; when MPA kill the target it will start another run if time
;; permit and still enough weapons and buoys
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(when (> searcherPK (random 1.0))
  (if (and (< ctime (MPAstruct-searchEndTime MPA))
          (< (MPAstruct-torpedoCount MPA) torpedosPerEngagement)
        (< (MPAstruct-buoyCount    MPA)      buoysPerSearch)
         )
    (endEngagement ctime MPA)
    (endMPAflight ctime MPA)
   )
  (ask  arget (die objectNumber))
   (return-from prosecuteTarget nil))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

 (if (MPAstruct-killedp MPA)(return-from prosecuteTarget nil))

(when (or (< (MPAstruct-torpedoCount MPA)
             torpedosPerEngagement)
             (< (MPAstruct-buoyCount MPA) buoysPerSearch))
  (endMPAflight ctime MPA)
  (ask target (endEngagement ctime))
   (return-from prosecuteTarget nil))

(expendBuoys   MPA   buoysPerEngagement)
(expendTorpedos MPA torpedosPerEngagement)
(when (and
      (> (ask target subEscapeMPAprob) (random 1.0))
      (> searcherPK (random 1.0)))
  (ask target (die objectNumber))

  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
  ;;
  ;; if time permits MPA will stay for another run
  ;;
  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
  (if (< ctime (MPAstruct-searchEndTime MPA))
```

```
      (endEngagement ctime MPA)
      (endMPAflight ctime MPA)
      )
     (return-from prosecuteTarget nil)
    )
   (ask target (endEngagement ctime))
   )
  )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (expendBuoys MPAsquadron) (MPA theNumber)
  (setf (MPAstruct-buoyCount MPA)
       (max 0 (- (MPAstruct-buoyCount MPA) theNumber)))
  (recordMOEevent (make-MOEeventStruct
                :eventID :buoyexpense
                :object1 MOEeventObjectIndex
                :object2 nil
                :data    theNumber))
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (expendTorpedos MPAsquadron) (MPA theNumber)
  (setf (MPAstruct-torpedoCount MPA)
       (max 0 (- (MPAstruct-torpedoCount MPA) theNumber)))
  (recordMOEevent (make-MOEeventStruct
                :eventID :MPAtorpedoExpense
                :object1 MOEeventObjectIndex
                :object2 nil
                :data    theNumber))
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; modified function
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defobfun (loseMPA MPAsquadron) (ctime MPA killer)
  (if (MPAstruct-killedp MPA)
     (return-from loseMPA nil))
  (setf (MPAstruct-killedp MPA) t)
```

```
(endMPAflight ctime MPA)
(let ((lside side))
  (ask MOEmonitor (updateRunningMOE :MPAkill lside)))
(recordMOEevent (make-MOEeventStruct
                 :eventID :attrition
                 :object1 MOEeventObjectIndex
                 :object2 killer)))
```

REFERENCES

[CA 91] Callahan, S. M., *Evaluation of Detection Modeling in ASSET*, Master's Thesis, Naval Postgraduate School, Monterey, CA, September 1991.

[CO 87] Cox, B. J., *Object-Oriented Programming: An Evolutionary Approach*, Addison-Wesley, Reading, MA. 1987.

[DU 86] Duff, C. B. *Designing an Efficient Language*, Byte, 11(8), pp211-224, 1986.

[GO 83] Goldberg, A. and Robson, D., *The Smalltalk-80: The Language and its Implementation*, Addison-Wesley, Reading, MA. 1983

[KI 87] Kim, T. G. and Aeigler, B. P., *The DEVS Formalism :Hierarchical, Modular System Specification in an Object Orientated Framework*, Proceedings of the 1987 Winter Simulation Conference, Atlanta, GA.

[RI 90] Richardson, H. R., Lent, S. C., and Stefanick, T.A., *ASW Systems Evaluation Tool (ASSET) Technical Documentation and User's Manual*, Metron Inc., March 1990.

[SC 87] Schmucker, K. J. and Cox, B. J., *Producer: A Tool for Translating Smalltalk-80 to Objective-C*, OOPSLA 1987 Conference Proceedings, Orlando, Fl.

[SH 91] Shaffer, R. M., *Evaluation of The MPA Detection and Allocation Models Utilized by ASSET*, Master's Thesis, Naval Postgraduate School, Monterey, CA. September 1991.

[ST 88] Stephen D. Roberts., Joe Heim., *A Perspective on Object-Oriented Simulation*, Proceeding of the 1988 Winter Simulation Conference.

[ST 90] Steele, Guy L., and others. *Common Lisp: The Language*. 2nd. edition. Digital Press, Maynard, Mass. 1990

[VE 91] Vebber, P. W., *An Examination of Target Tracking in ASSET*, Master's Thesis, Naval Postgraduate School, Monterey, CA. September 1991.

[WA 91] Wagner, D. H., "Cumulative Detection Probability," draft of Chapter 5 for 3rd ed. of Naval Operations Analysis, June 1991.

[ZE 84] Zeigler, B. P., *Multifacetted Modelling and Discrete Event Simulation*, Academic Press, London and Orlando, FL. 1984.

INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Technical Information Center Cameron Station Alexandria, VA. 22304-6145	2
2. Library, Code 052 Naval Postgraduate School Monterey, CA. 93943-5002	2
3. Chief of Naval Operations Antisubmarine Warfare Division (OP-71B2) Wachington D.C. 20350-2000 Attn: CDR Moses	1
4. Metron, Inc. 11911 Freedom Drive Suite 800 Reston, VA. 22090-5603 Attn: Dr. Richardson	1
5. Dr. Yuh-jeng Lee, Code CS/Le Department of Computer Science Naval Postgraduate School Monterey, CA. 93943-5100	5
6. Dr. James N. Eagle, Code OR/ER Department of Operations Research Naval Postgraduate School Monterey, CA. 93943-5100	1
7. LT Commander Peng-tso Chang 37, Lane 142, Ho-kuang St. Tso-ying, Kaohsiung Taiwan, Republic of China	3